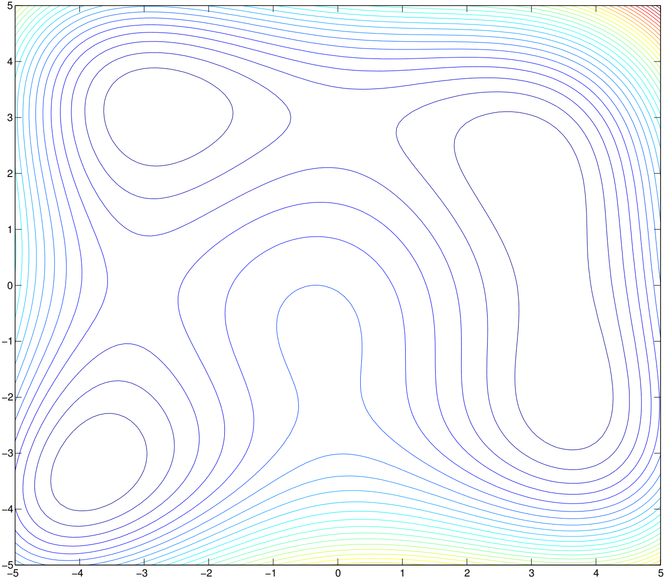
Exploiting Problem-Specific Knowledge and Computational Resources in Derivative-Free Optimization

Jeffrey Larson

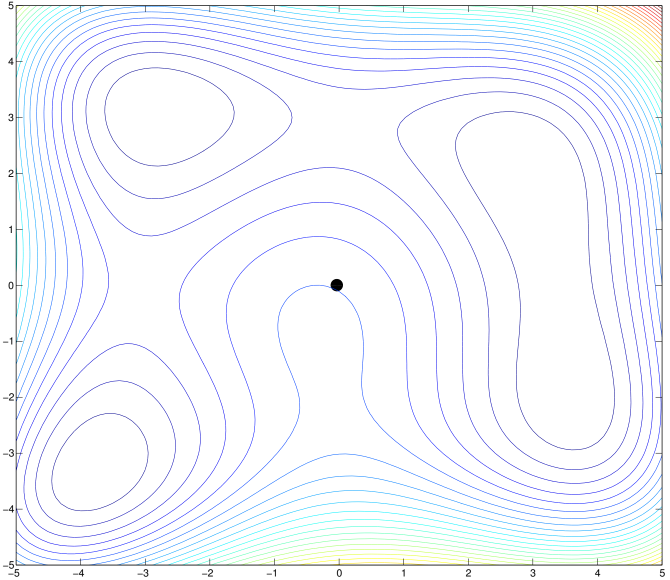
Argonne National Laboratory

September 8, 2015

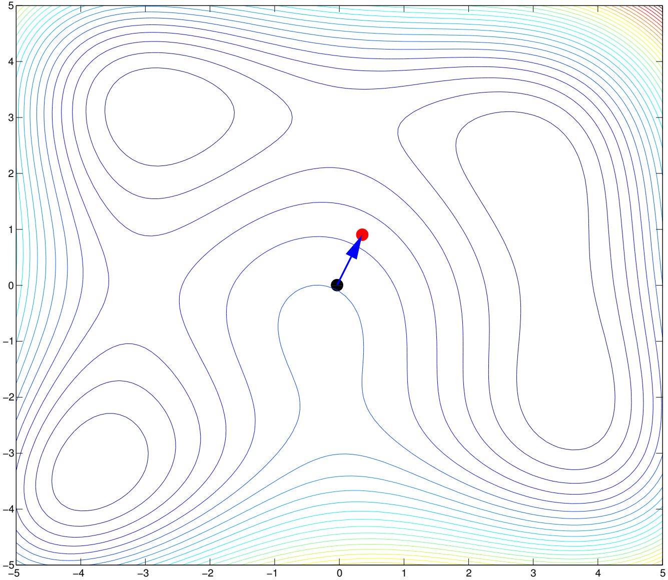
Line Search



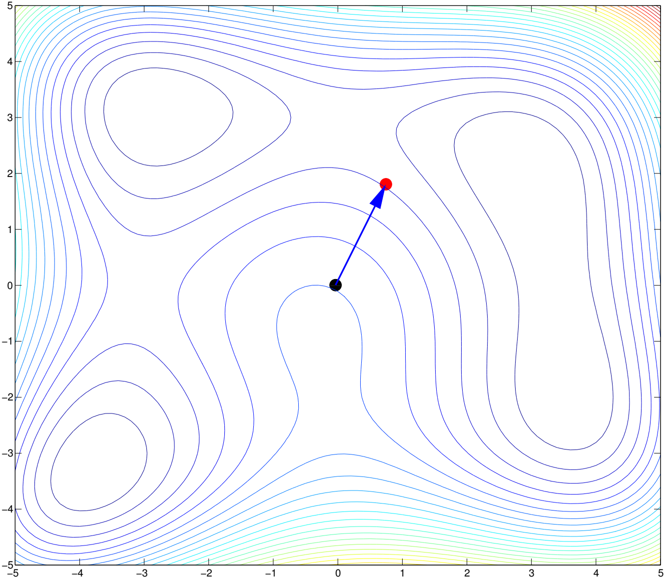
Line Search



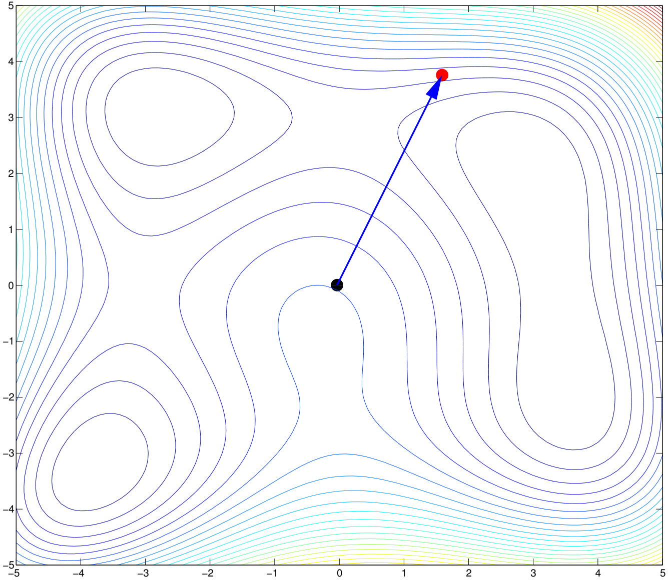
Line Search



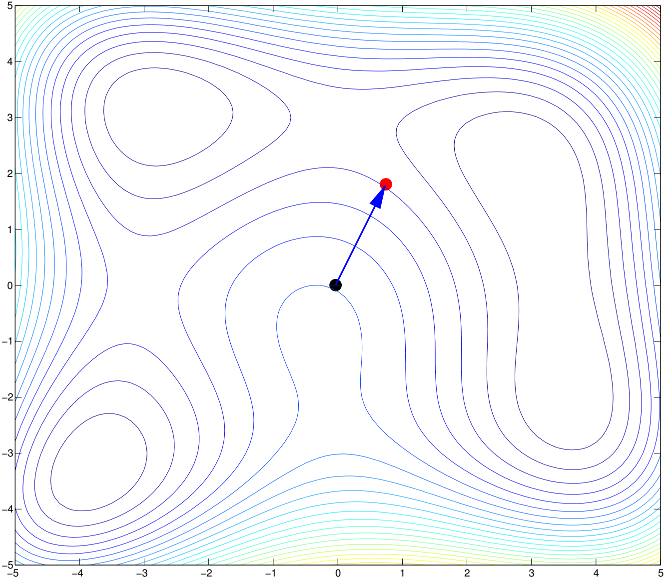
Line Search



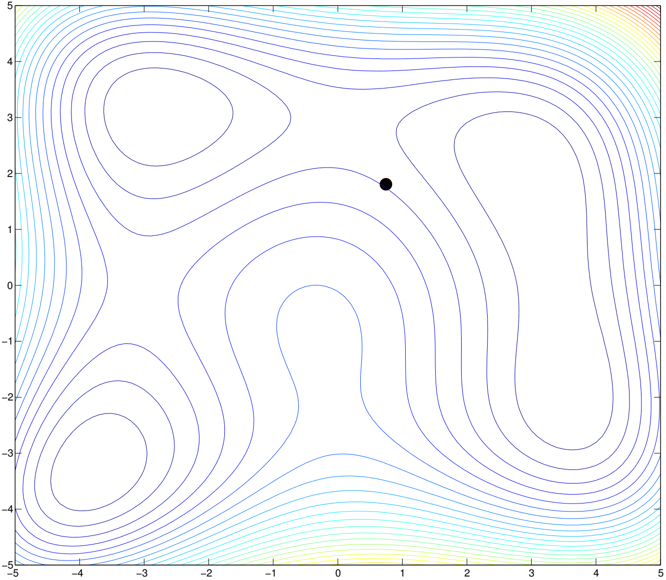
Line Search



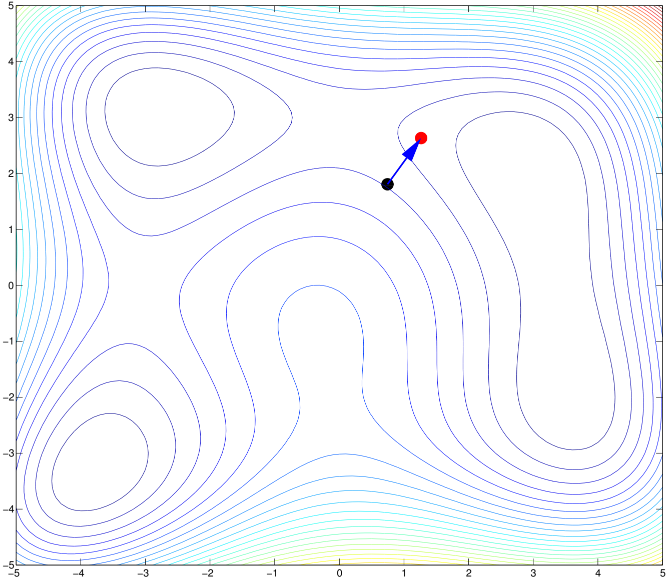
Line Search



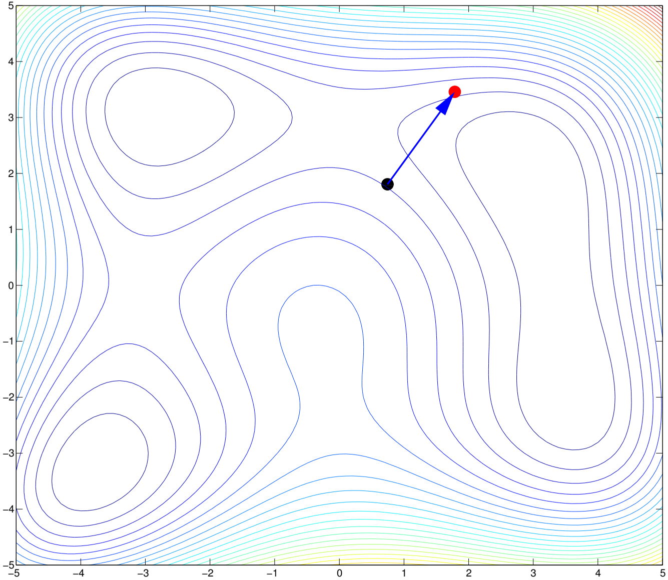
Line Search



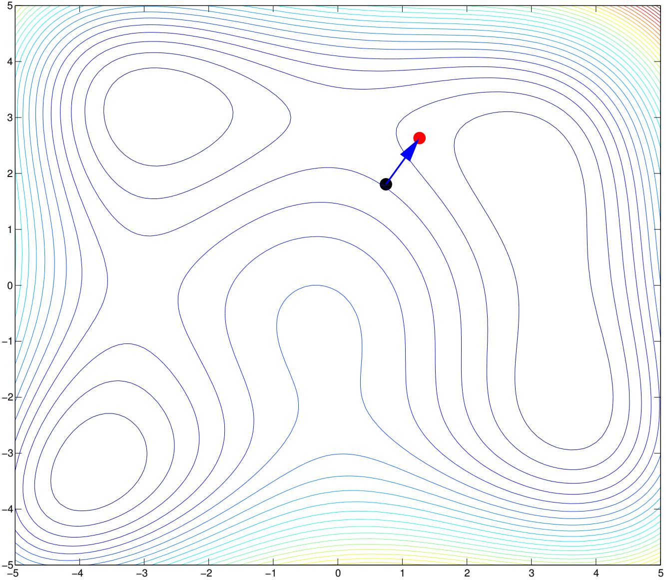
Line Search



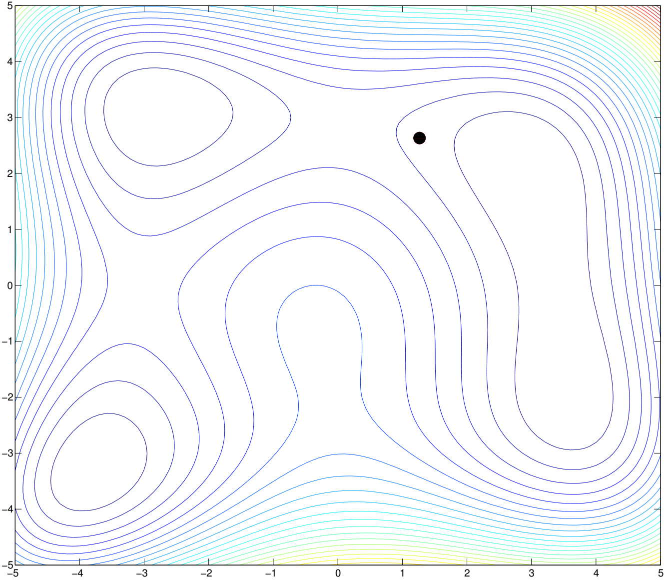
Line Search



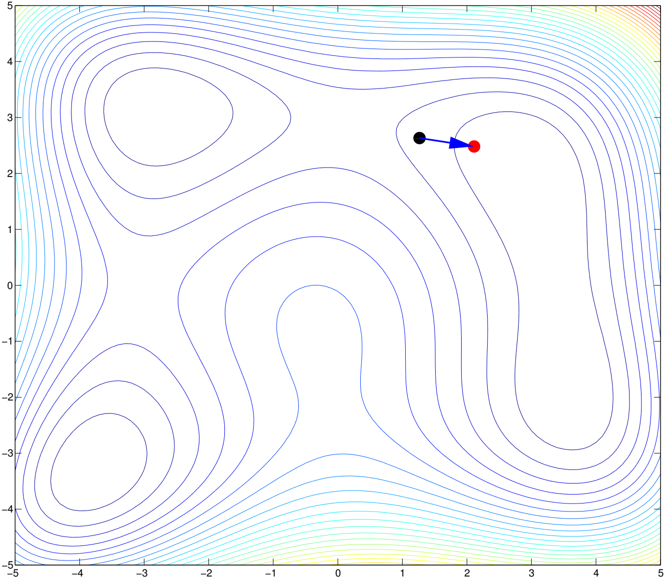
Line Search



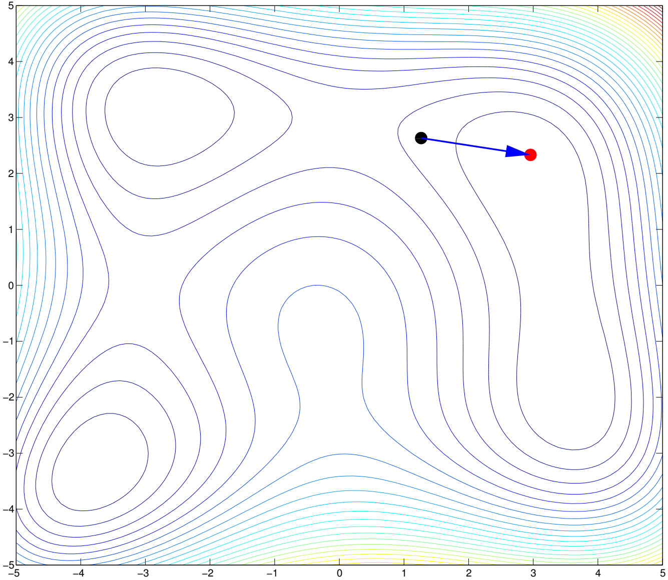
Line Search



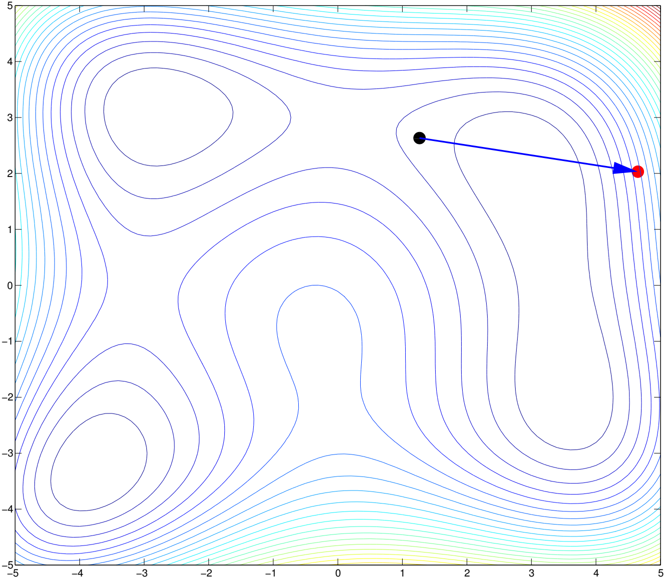
Line Search



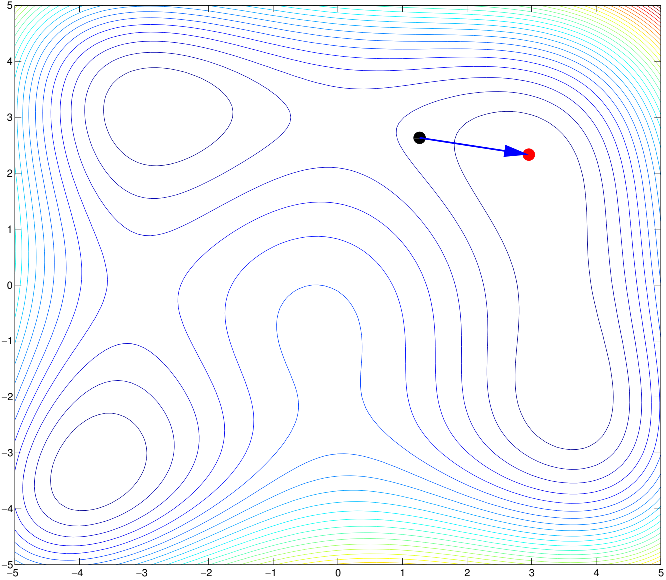
Line Search



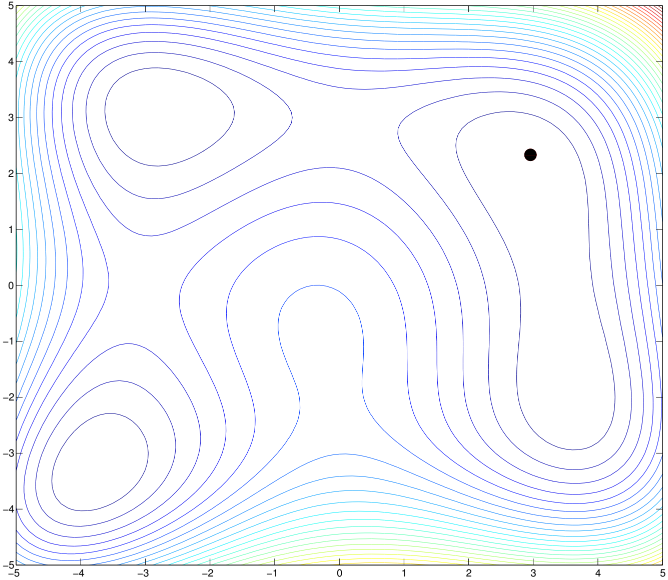
Line Search



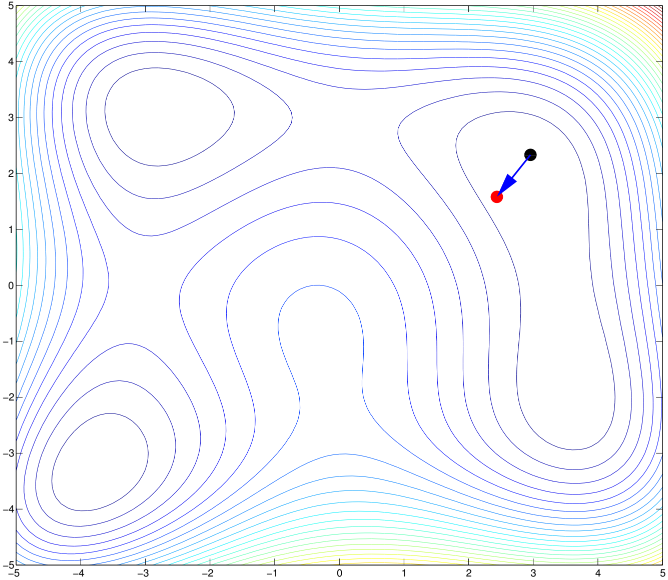
Line Search



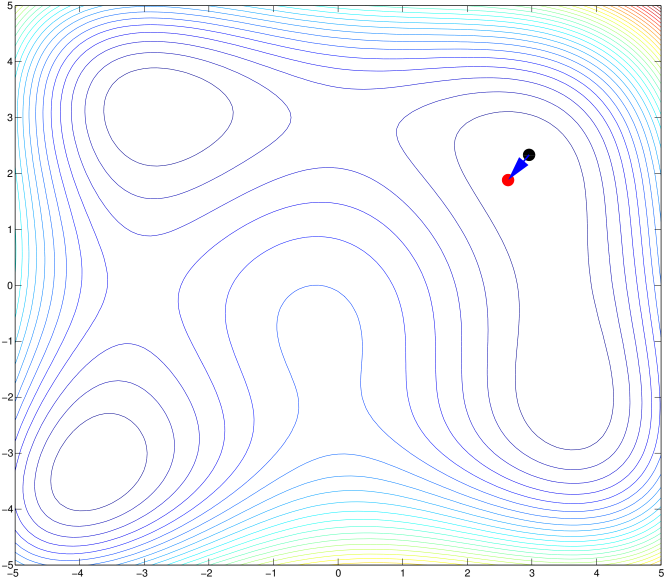
Line Search



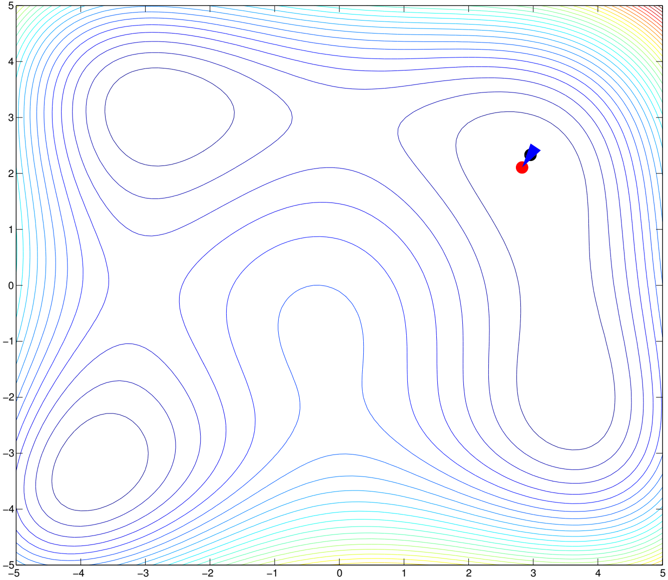
Line Search



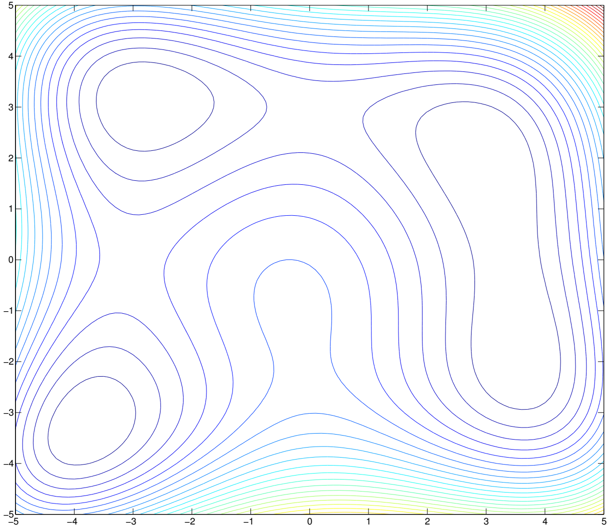
Line Search



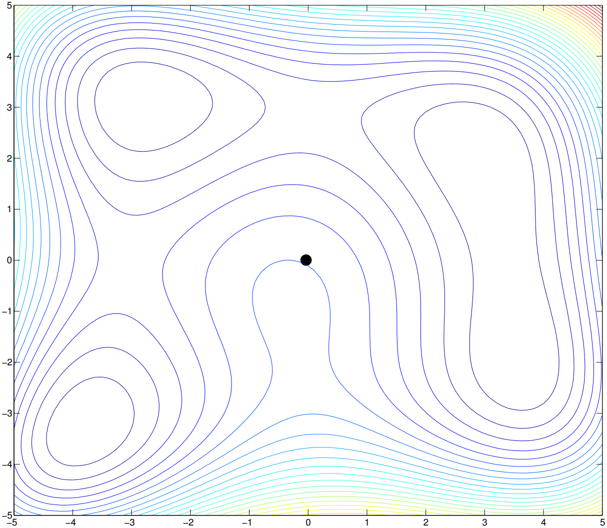
Line Search



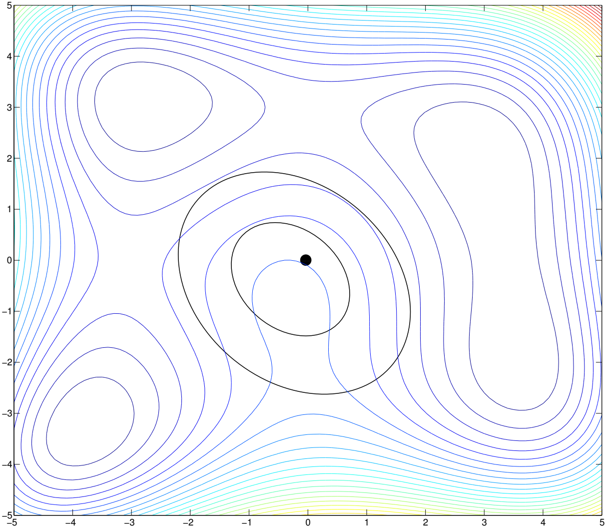
Trust Region Methods



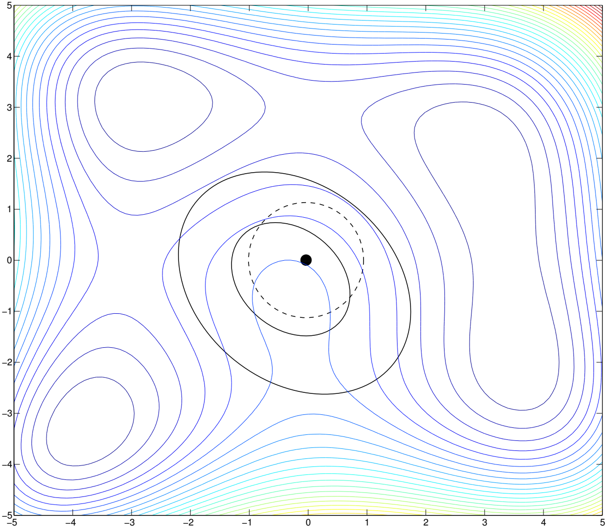
Trust Region Methods



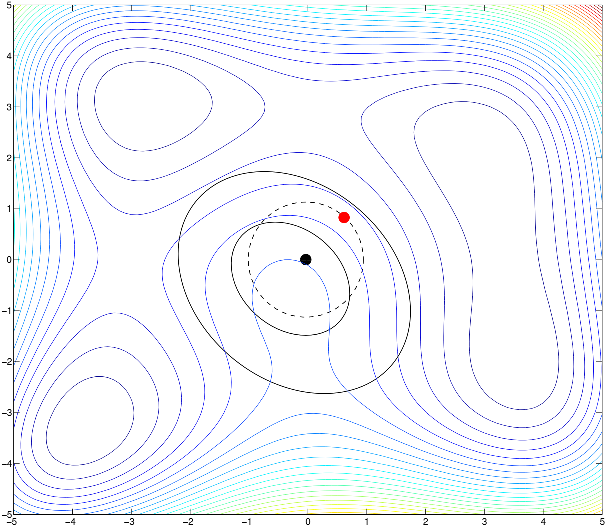
Trust Region Methods



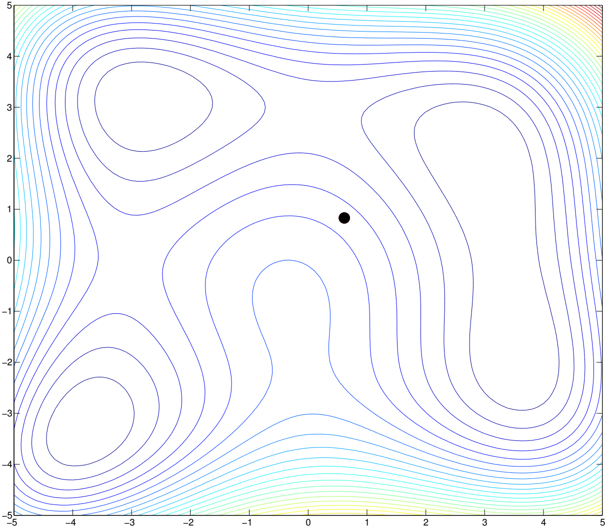
Trust Region Methods



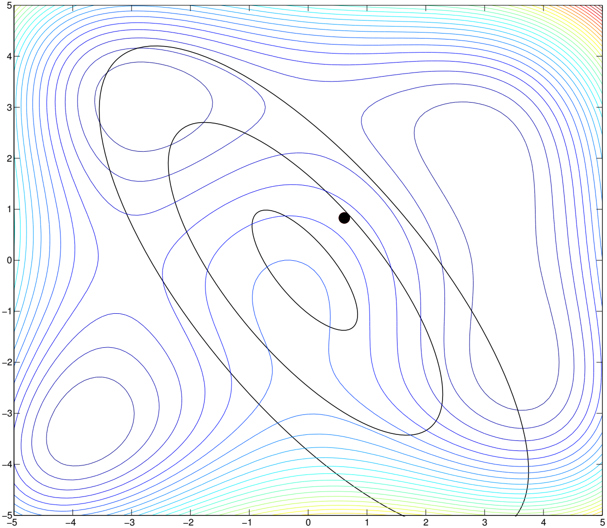
Trust Region Methods



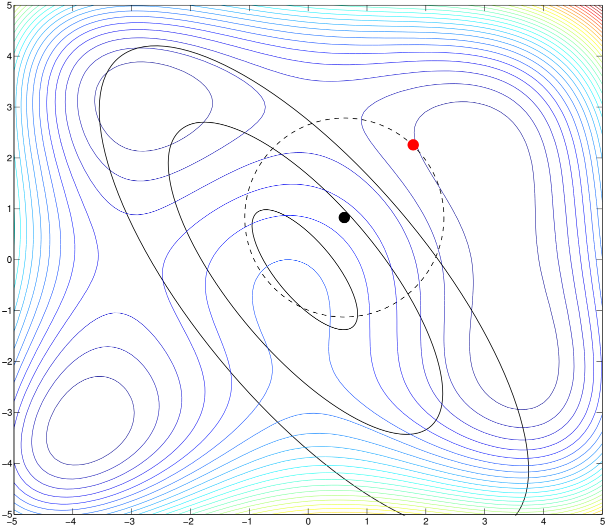
Trust Region Methods



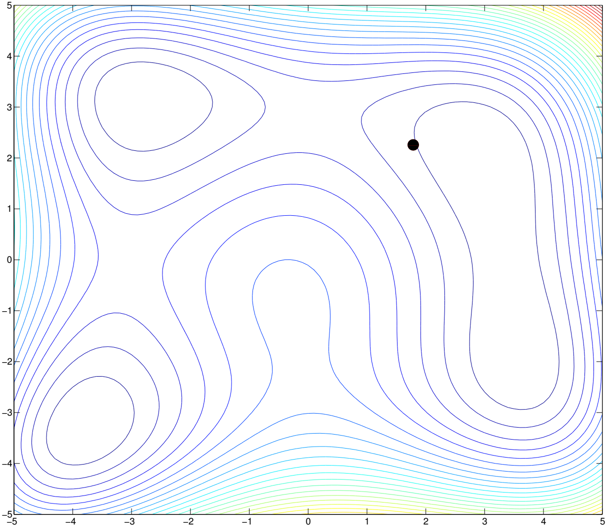
Trust Region Methods



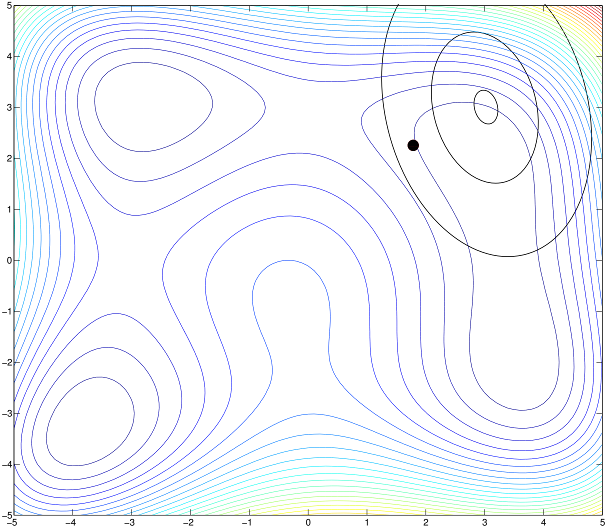
Trust Region Methods



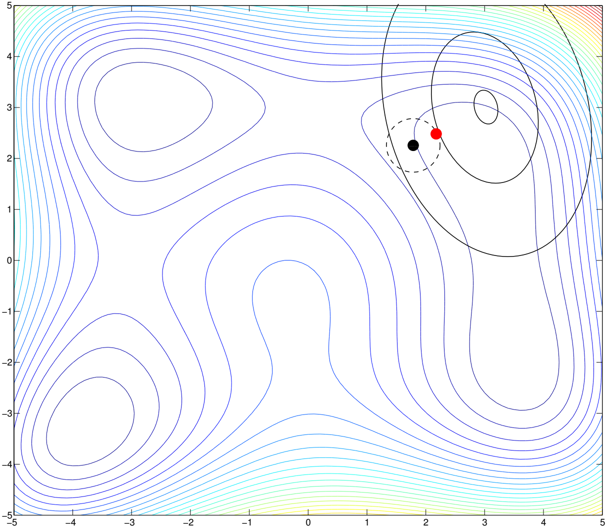
Trust Region Methods



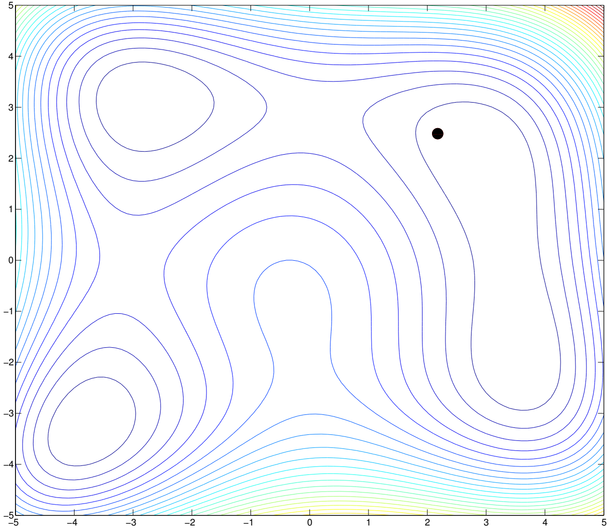
Trust Region Methods



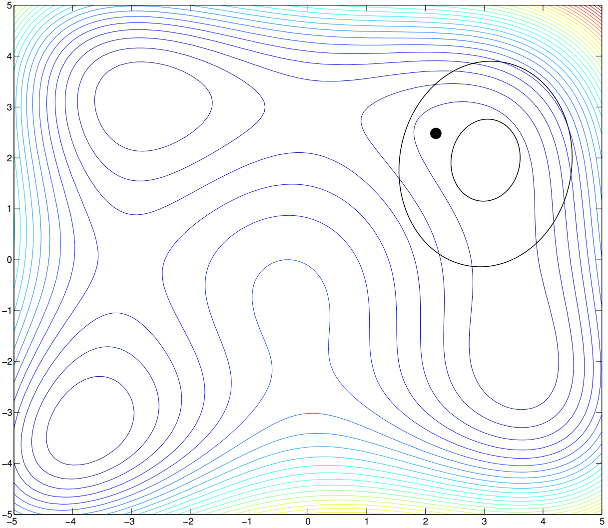
Trust Region Methods



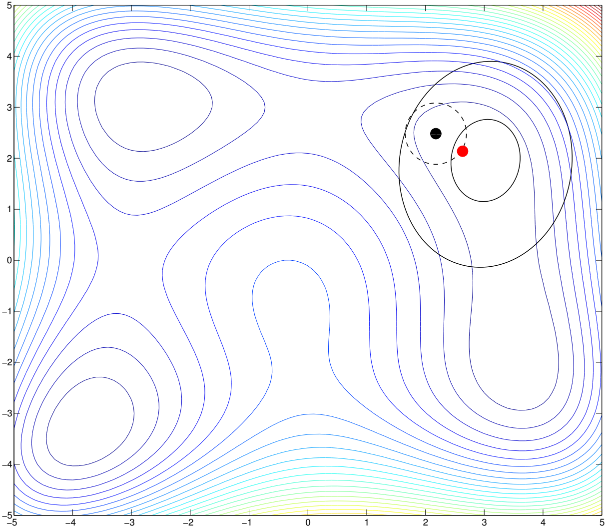
Trust Region Methods



Trust Region Methods



Trust Region Methods



Problem setup

$$\underset{x}{\text{minimize}}\ f(x; S(x))$$

$$\textbf{subject to: } x \in \mathcal{D} \subset \mathbb{R}^n$$

where the objective f depends on the output(s) from a simulation $S(x)$.

Problem setup

$$\underset{x}{\text{minimize}}\ f(x; S(x))$$

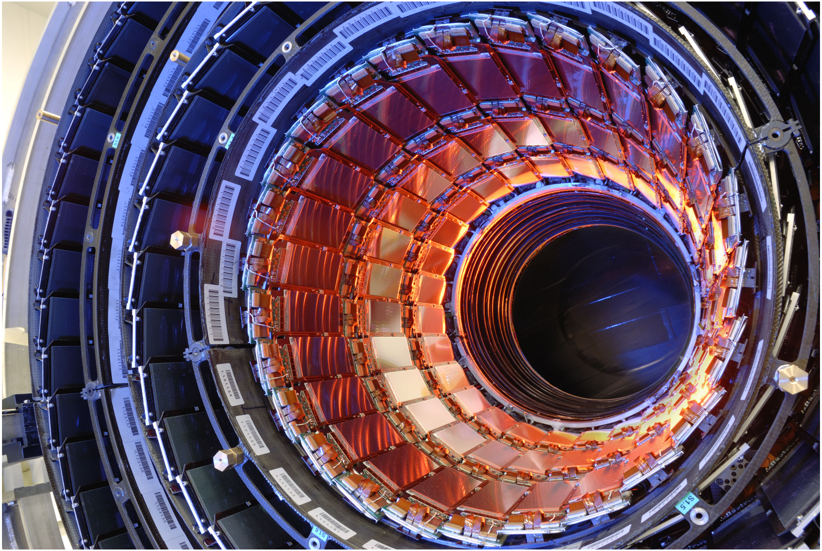
$$\text{subject to: } x \in \mathcal{D} \subset \mathbb{R}^n$$

where the objective f depends on the output(s) from a simulation $S(x)$.

- Derivatives of S may not be available
- Constraints defining $\mathcal{D}$ may or may not depend on S
- The dimension n is small
- Evaluating S is expensive
- f and/or S may be noisy. If the noise is stochastic,

$$\underset{x}{\text{minimize}}\ \mathbb{E}\left[\bar{f}(x)\right].$$

Motivation

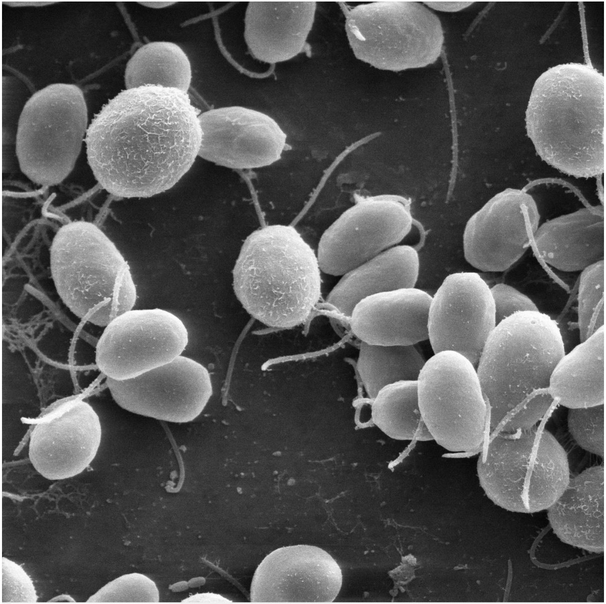


LHC: $4.75B to build; $1B to operate; $0.25B computation

Motivation

Mira: \$180M to build; \$4M to operate; 3.9 Megawatts

Motivation



10 µm Chlamydomonas6.001 5/ 3/ 0 REMF 3000X

Motivation

Initial approaches

- Grid over the domain (easily parallelizable)

Initial approaches

- Grid over the domain (easily parallelizable)
- Random sampling (easily parallelizable)

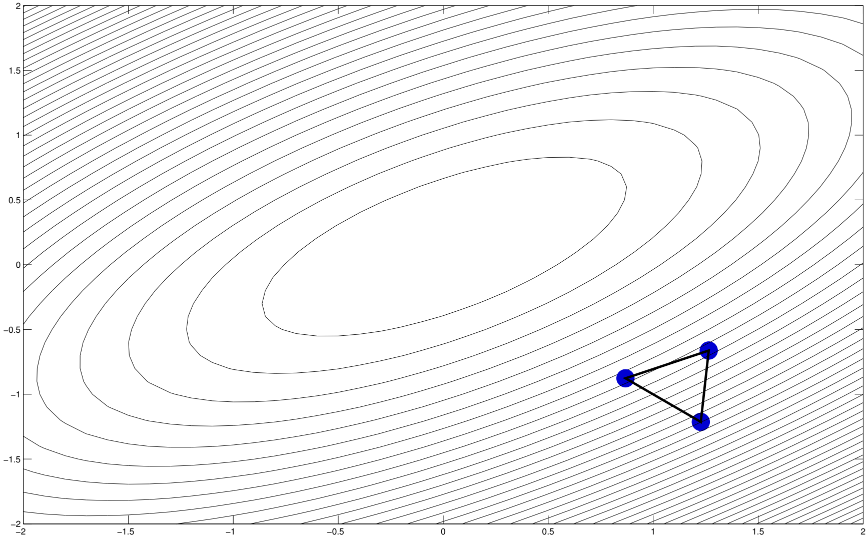
Initial approaches

- Grid over the domain (easily parallelizable)
- Random sampling (easily parallelizable)
- Evolutionary Algorithms (many are parallelizable)

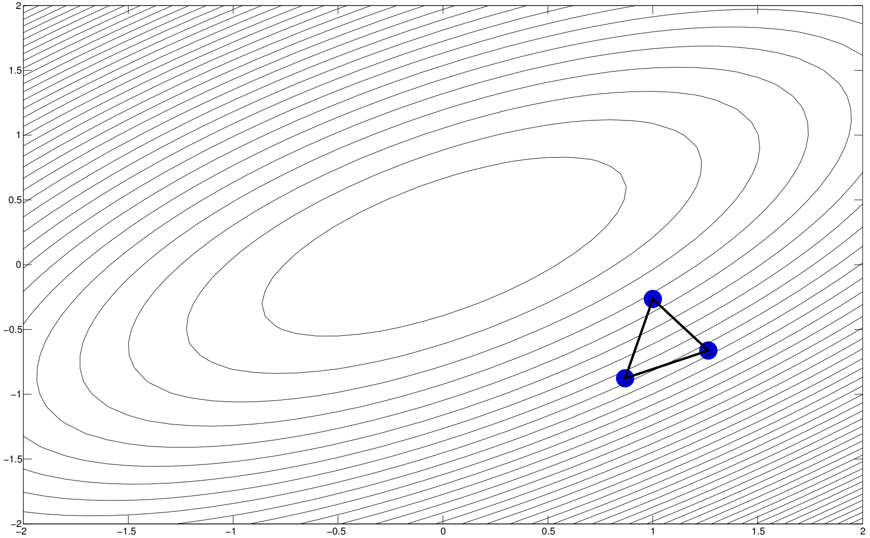
Initial approaches

- Grid over the domain (easily parallelizable)
- Random sampling (easily parallelizable)
- Evolutionary Algorithms (many are parallelizable)
  - Genetic Algorithm
  - Simulated Annealing
  - Particle Swarm
  - Ant Colony Optimization
  - Bee Colony Optimization
  - Cuckoo Search
  - Bacterial Colony Optimization
  - Grey Wolf Optimization
  - Firefly Optimization
  - Harmony Search
  - River Formation Dynamics

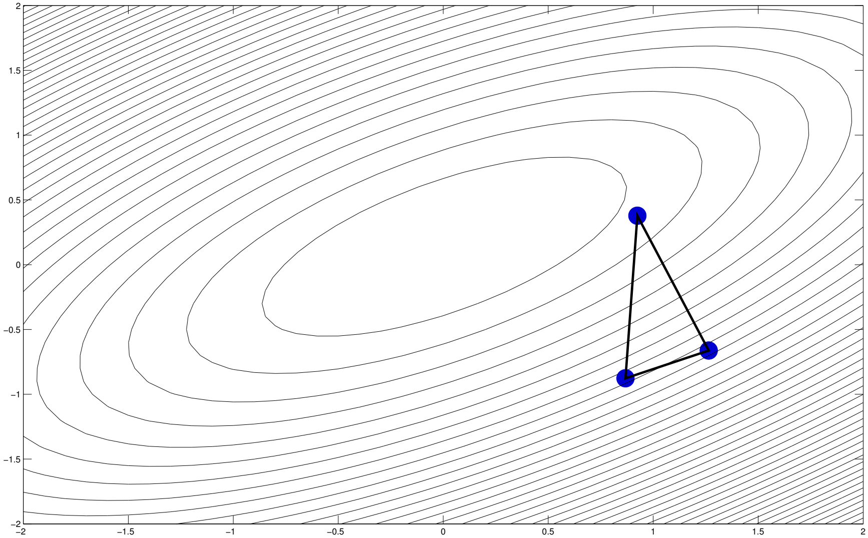
Nelder-Mead



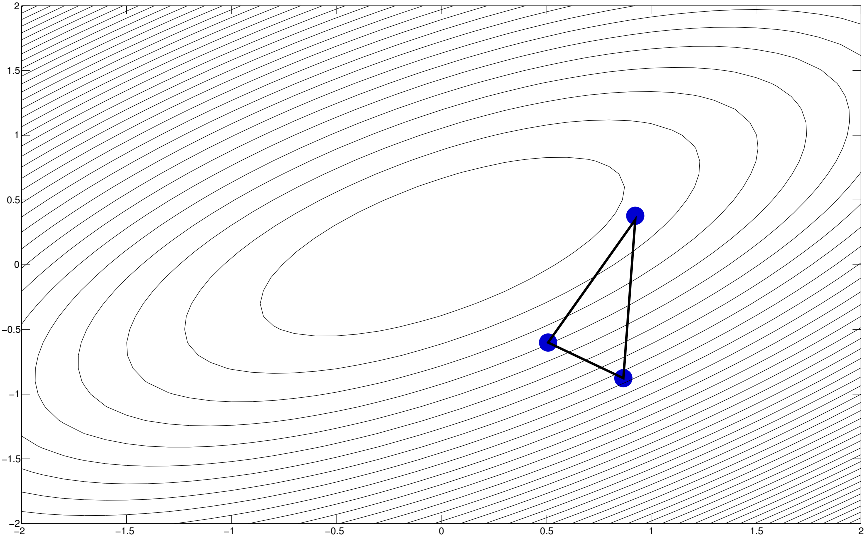
Nelder-Mead



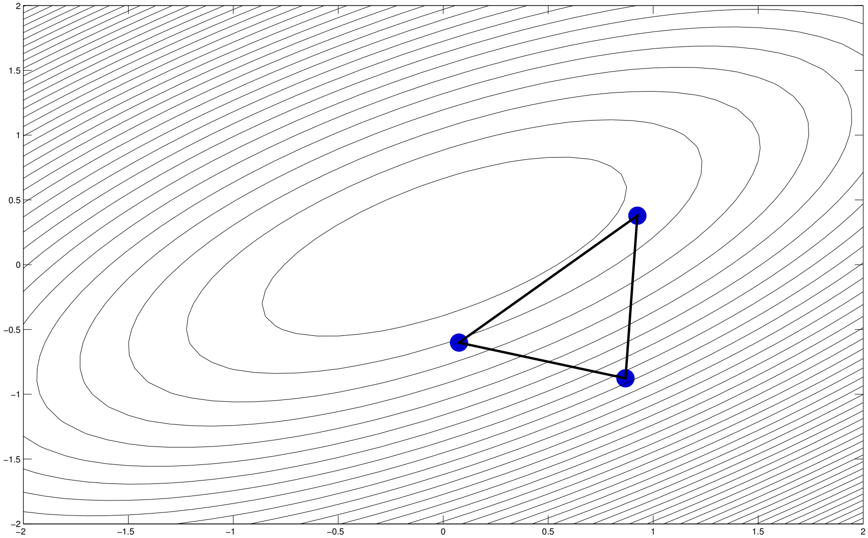
Nelder-Mead



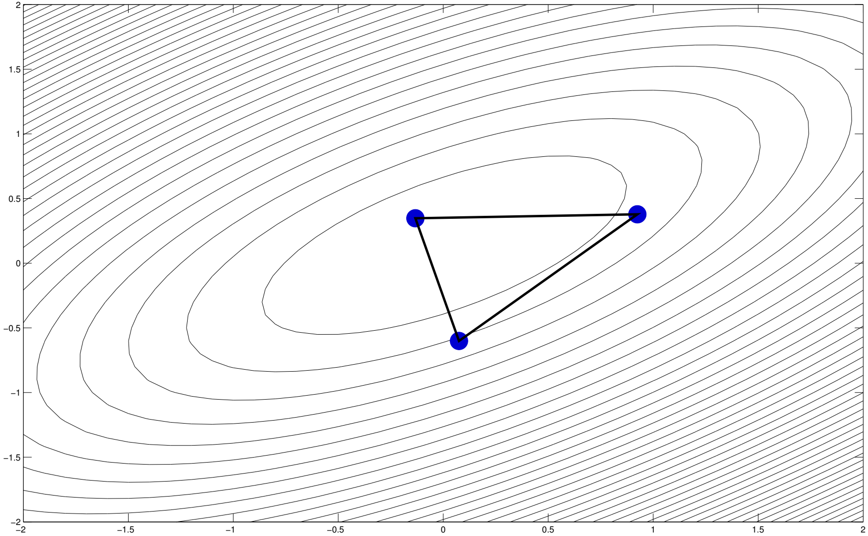
Nelder-Mead



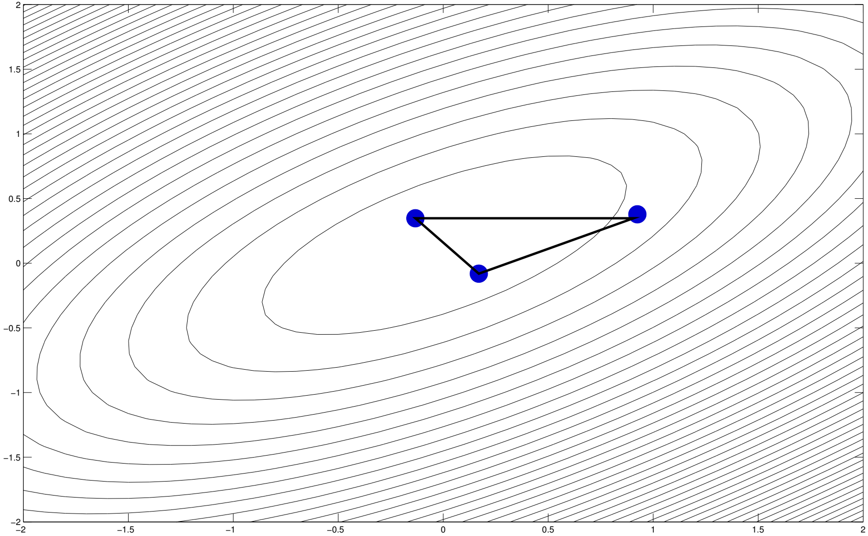
Nelder-Mead



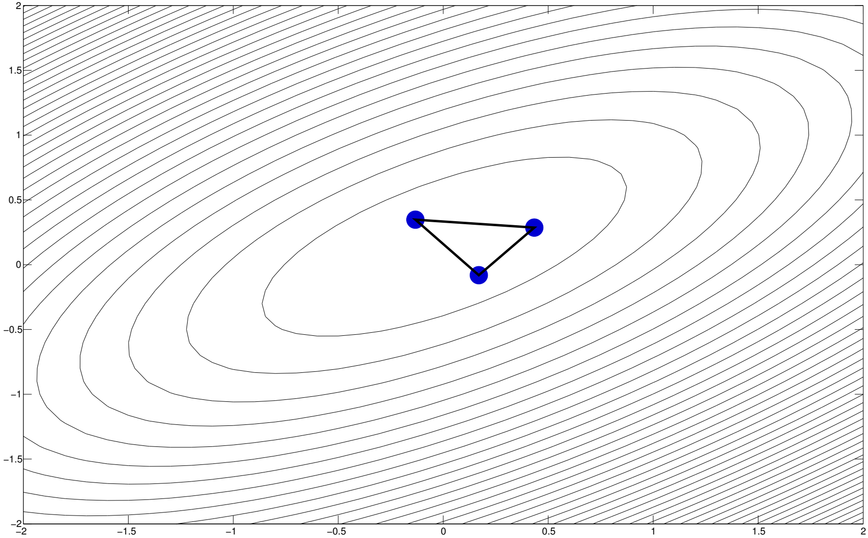
Nelder-Mead



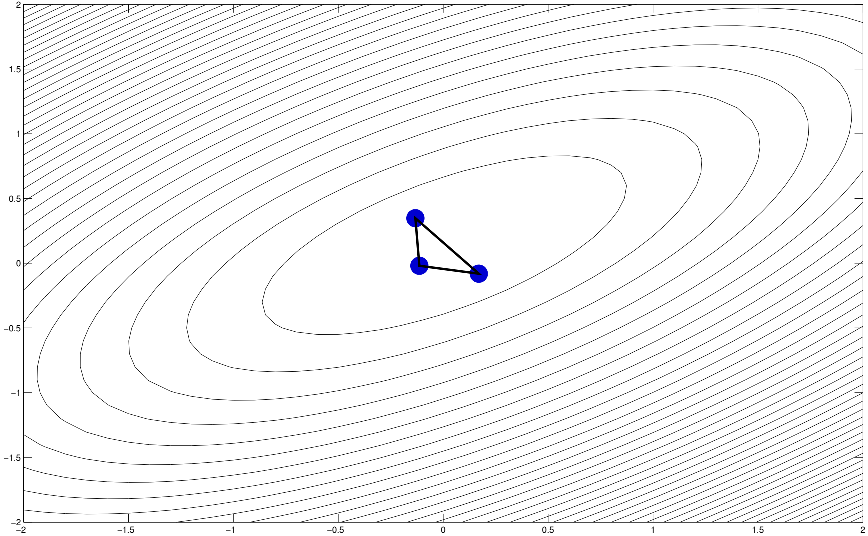
Nelder-Mead



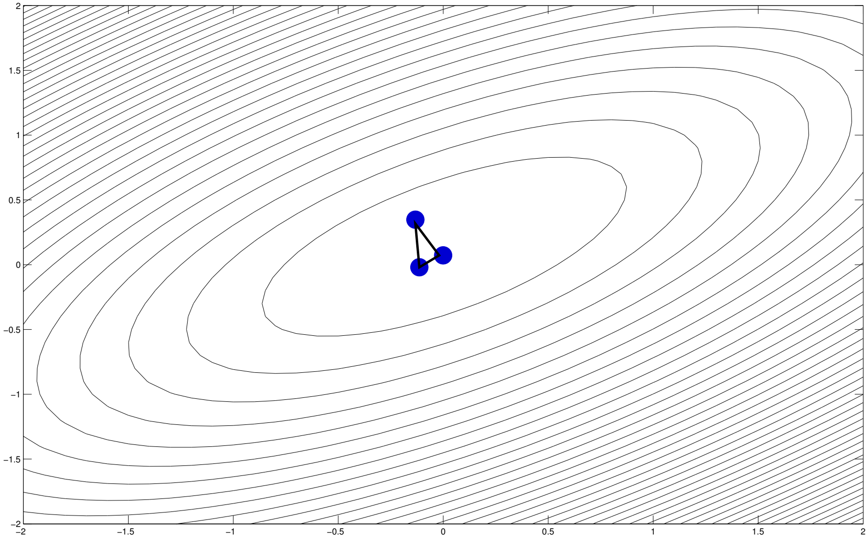
Nelder-Mead



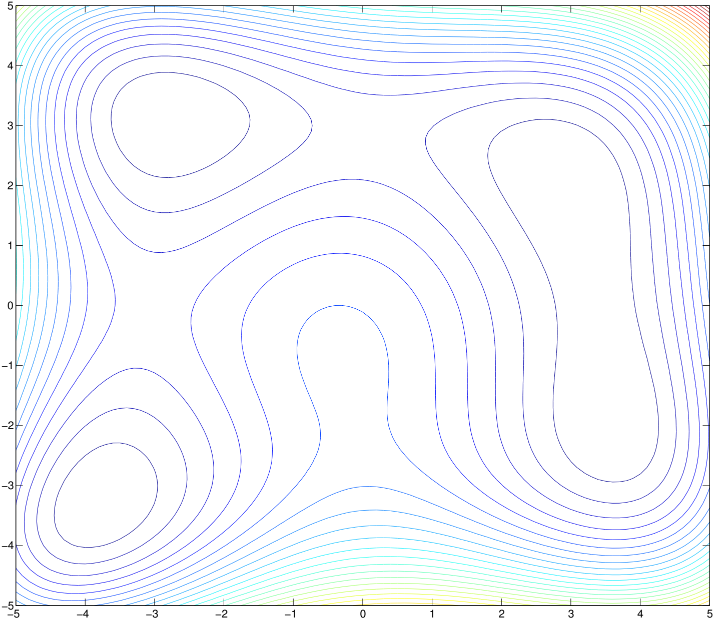
Nelder-Mead



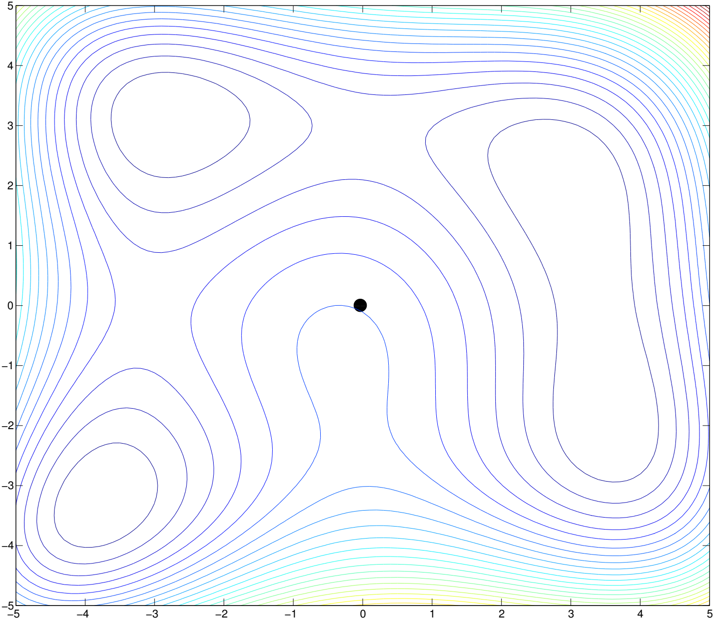
Nelder-Mead



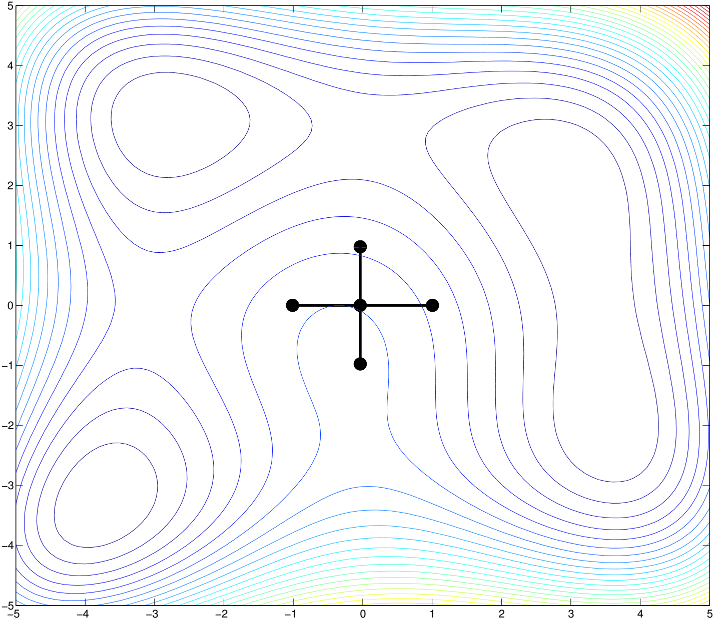
Coordinate Search



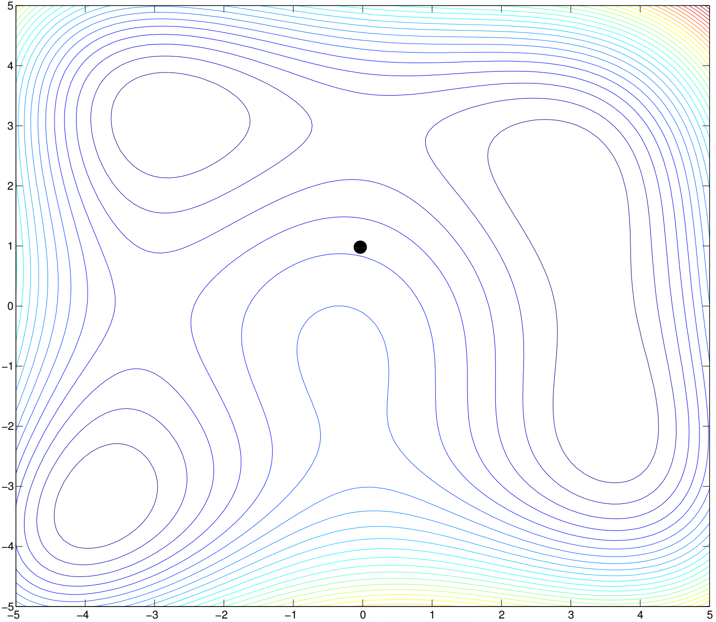
Coordinate Search



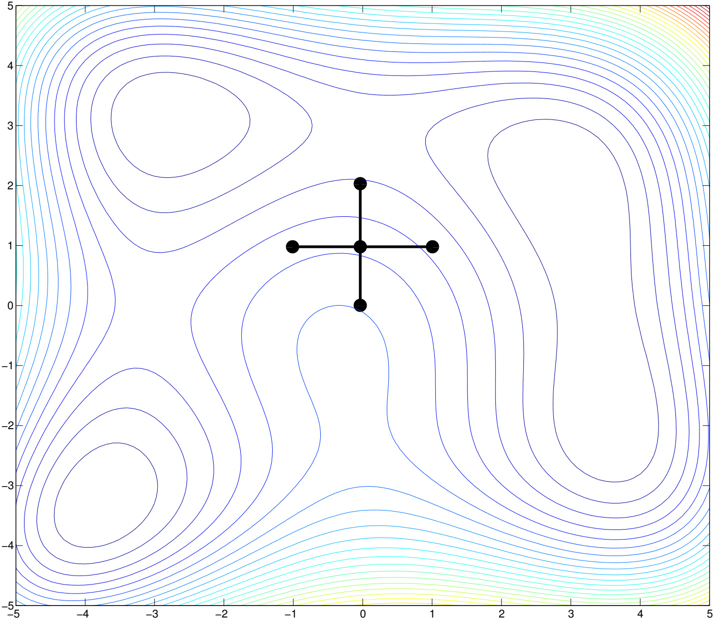
Coordinate Search



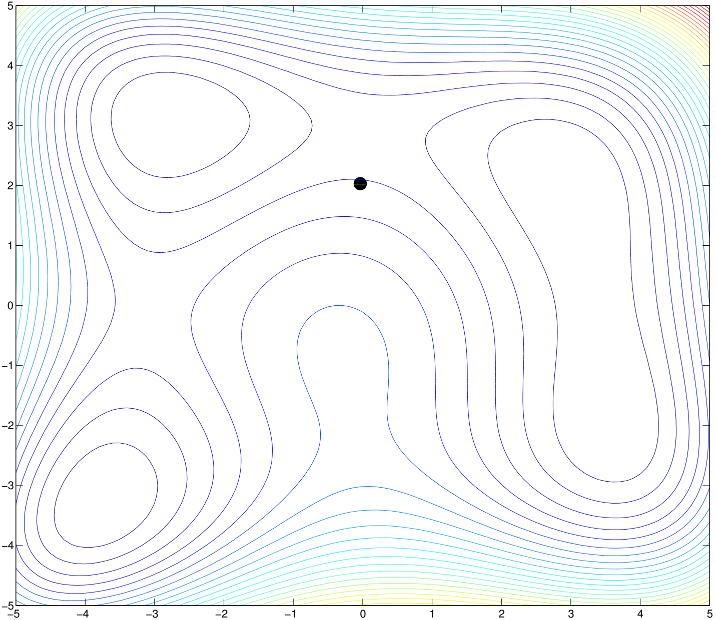
Coordinate Search



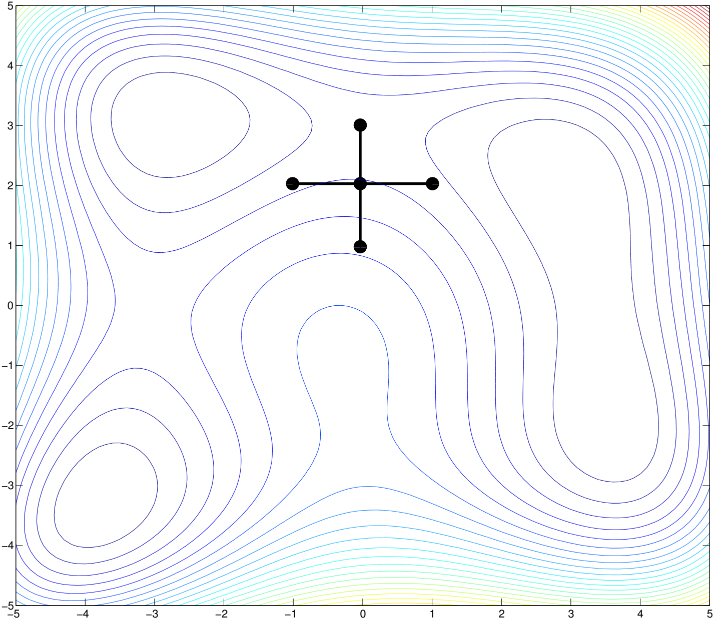
Coordinate Search



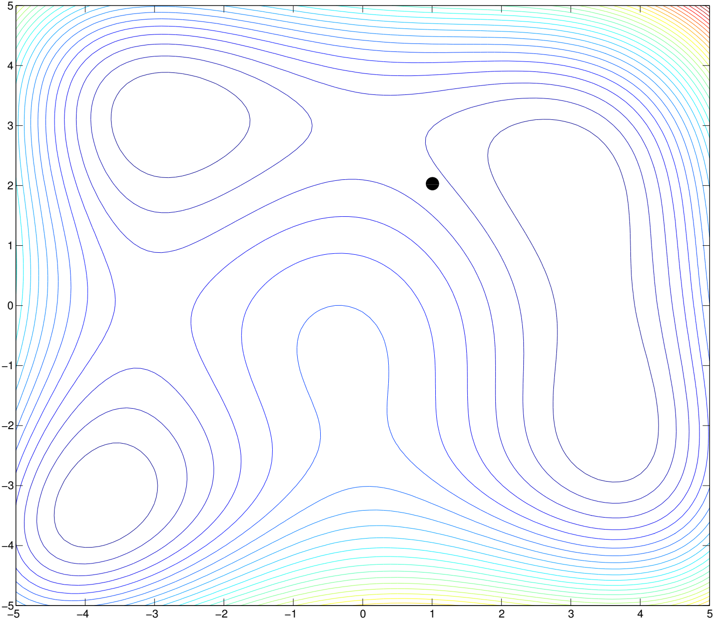
Coordinate Search



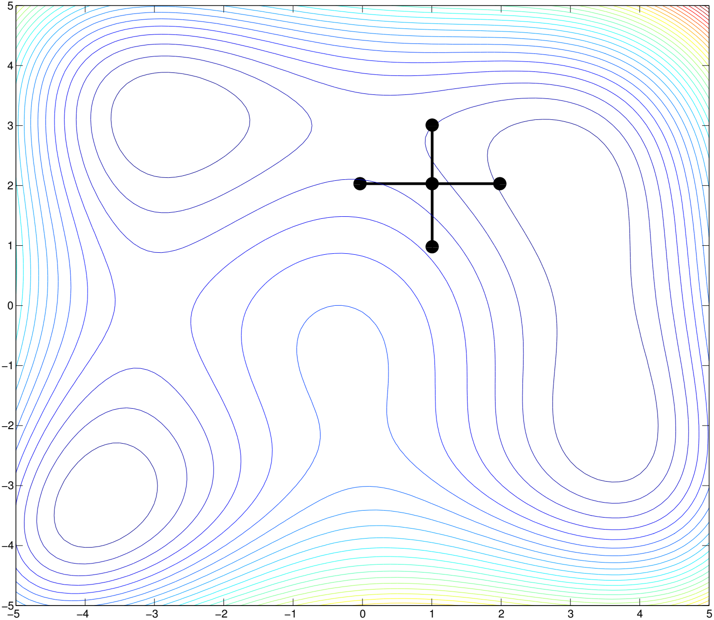
Coordinate Search



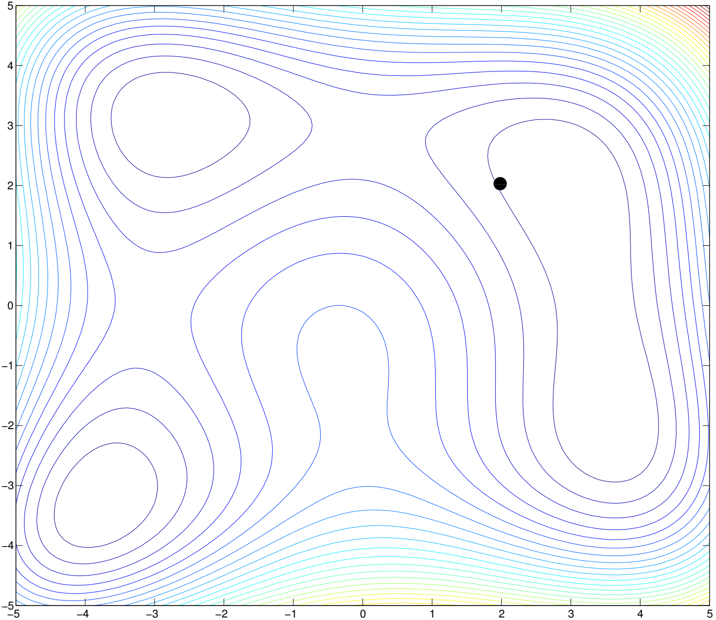
Coordinate Search



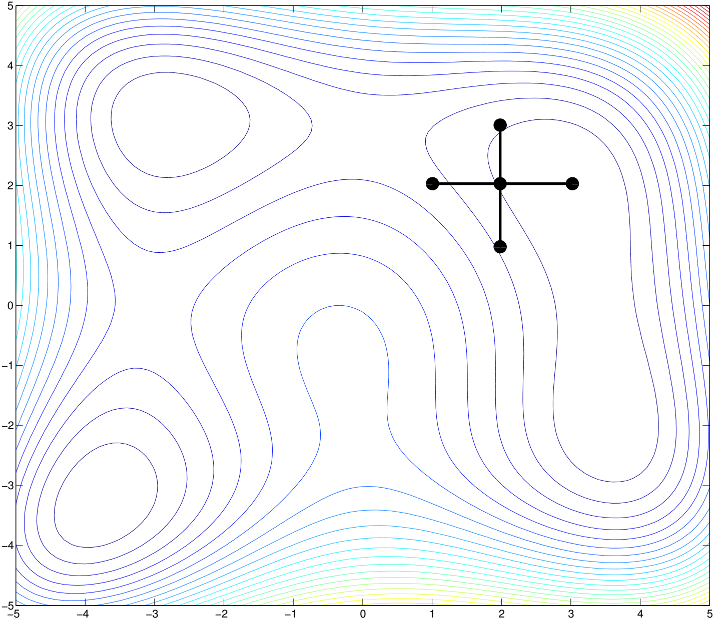
Coordinate Search



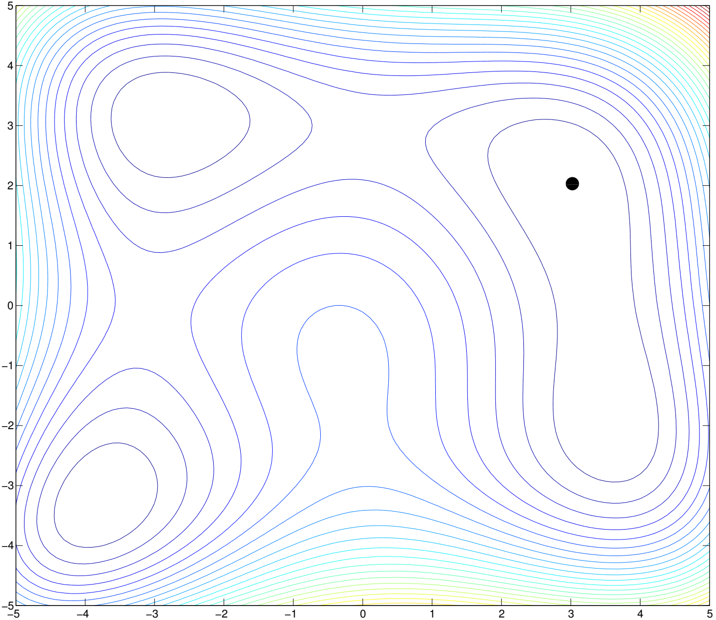
Coordinate Search



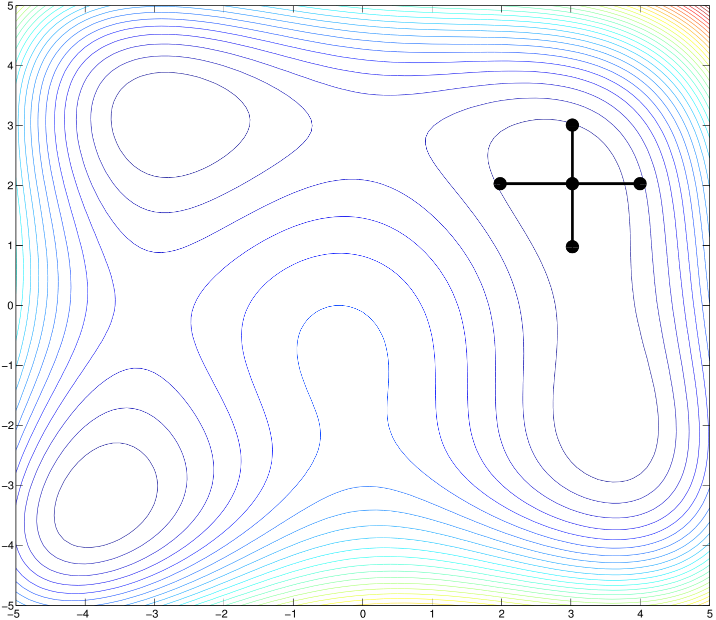
Coordinate Search



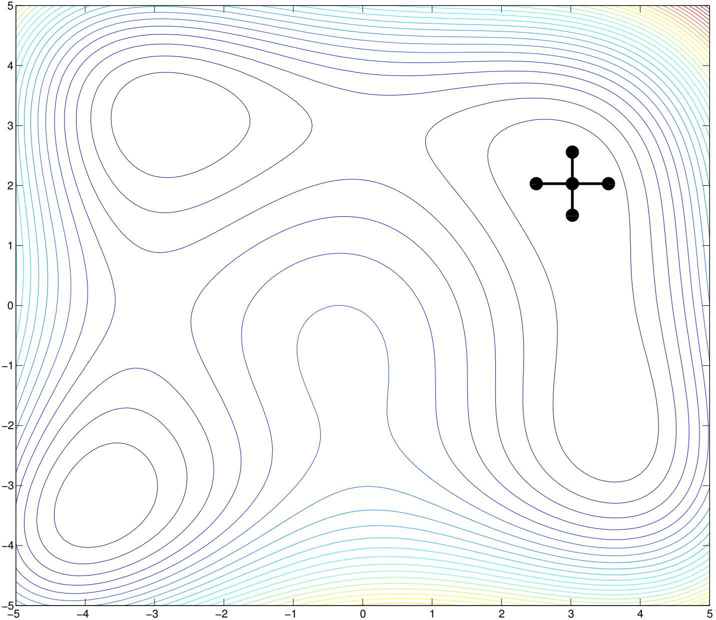
Coordinate Search



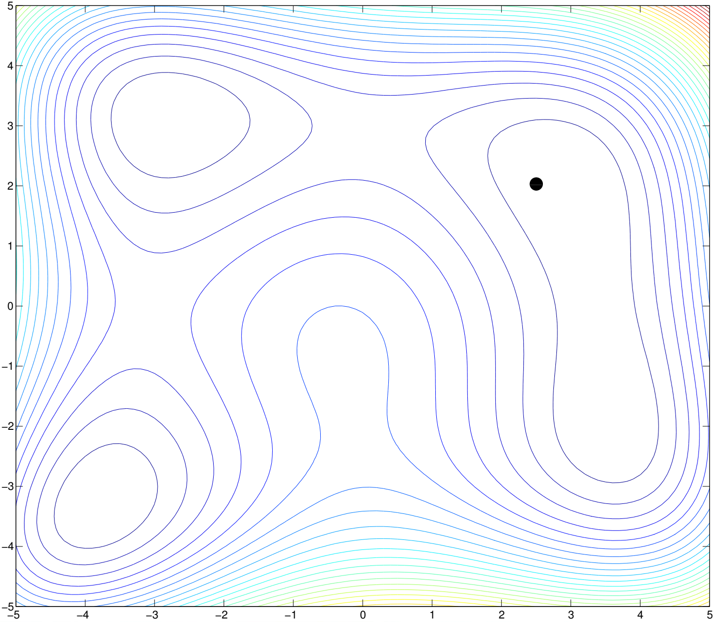
Coordinate Search



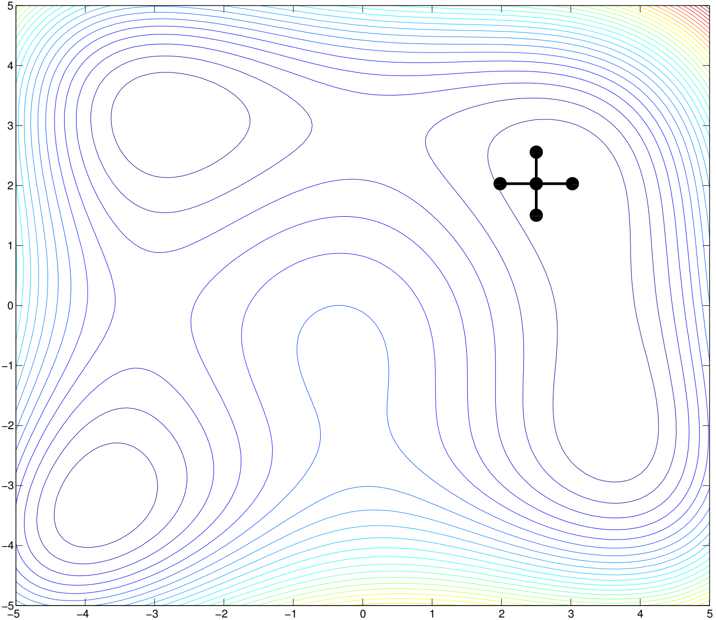
Coordinate Search



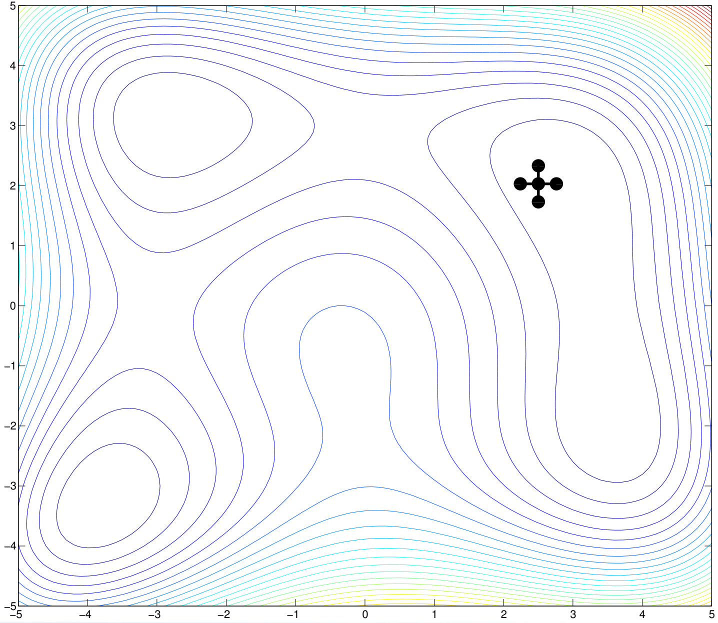
Coordinate Search



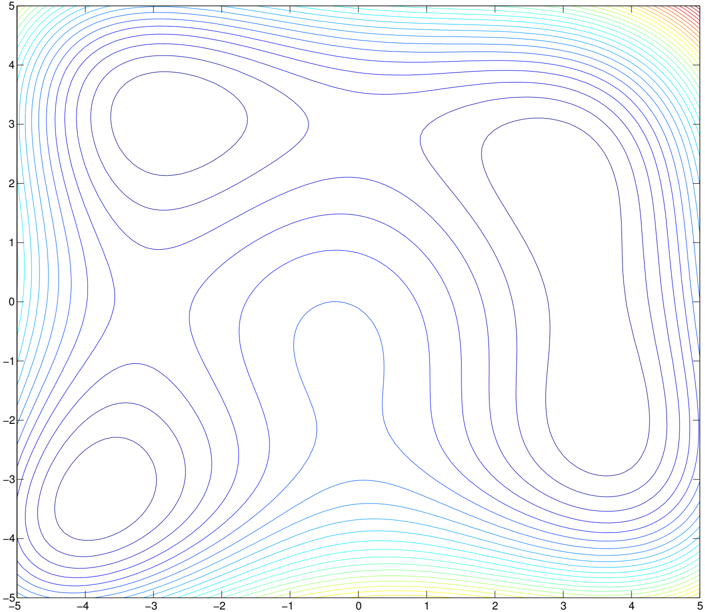
Coordinate Search



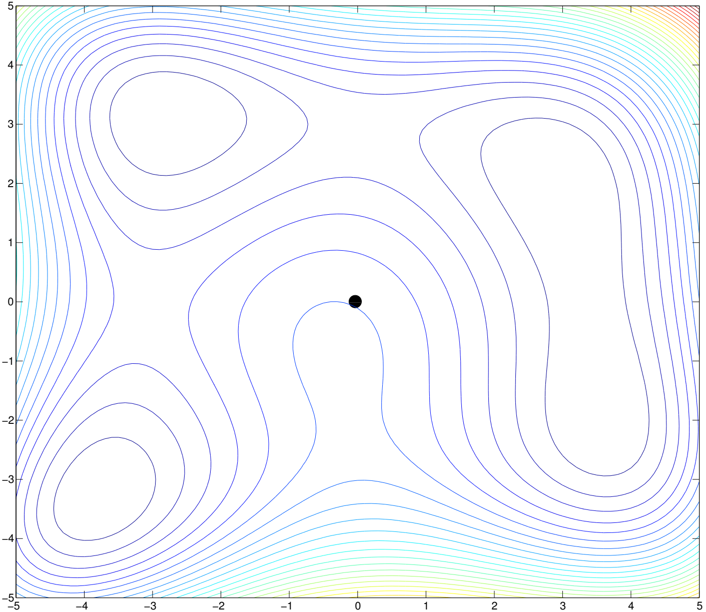
Coordinate Search



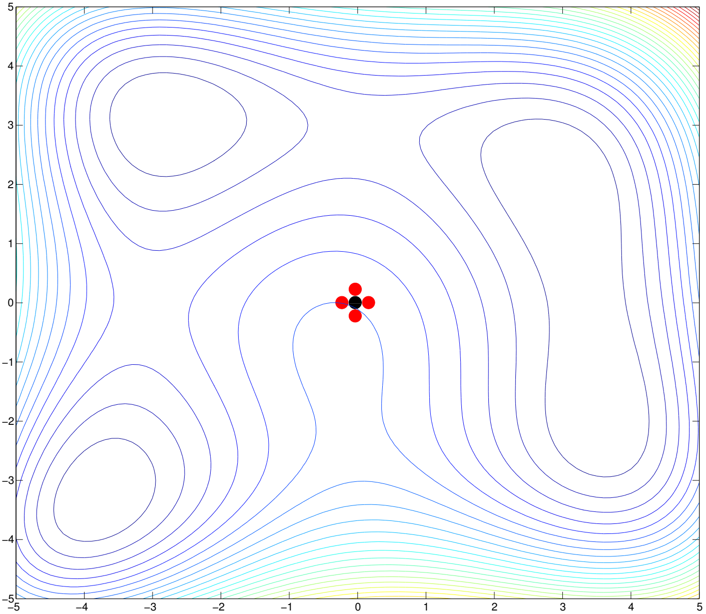
Approximate Gradients



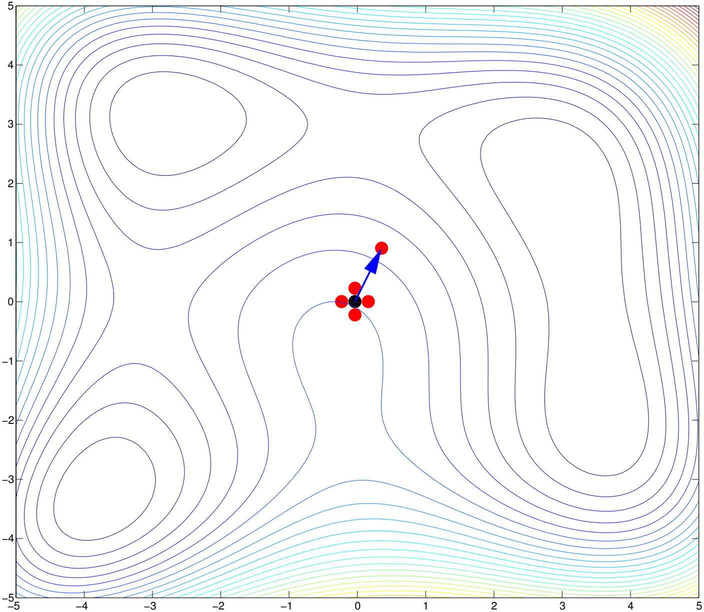
Approximate Gradients



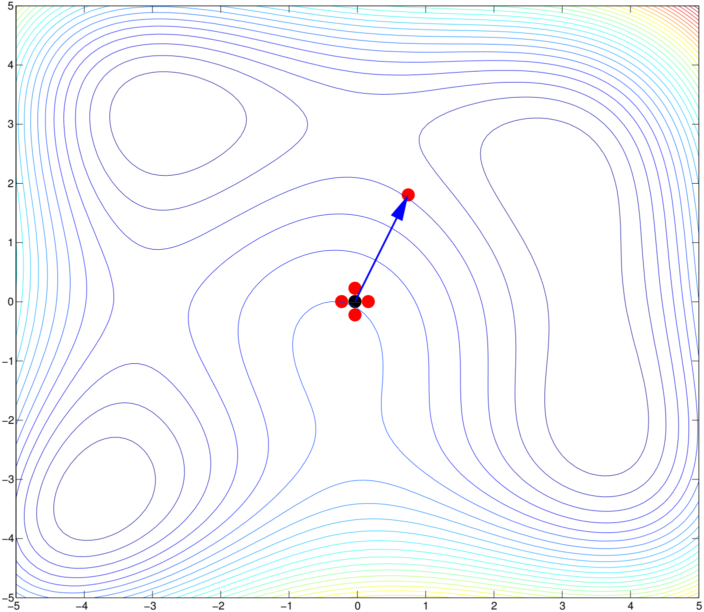
Approximate Gradients



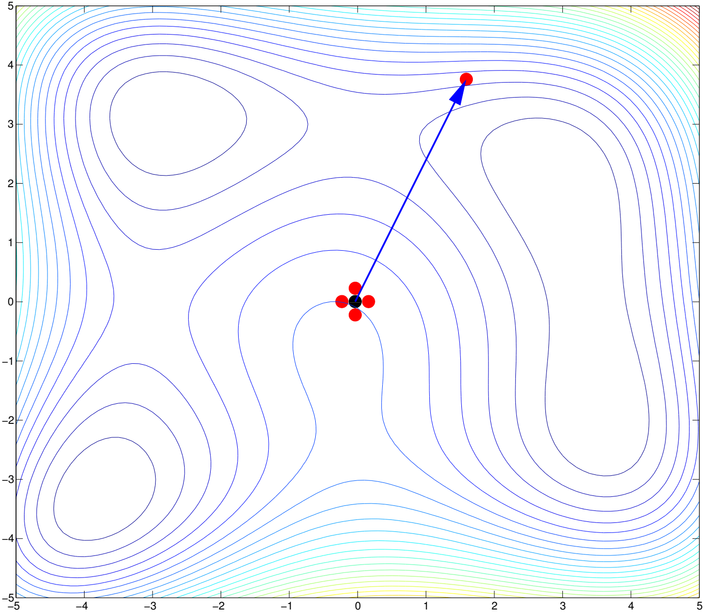
Approximate Gradients



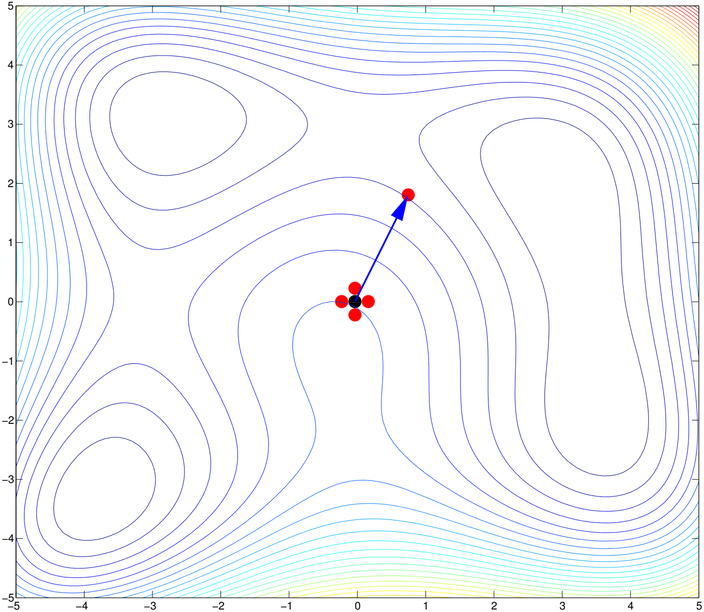
Approximate Gradients



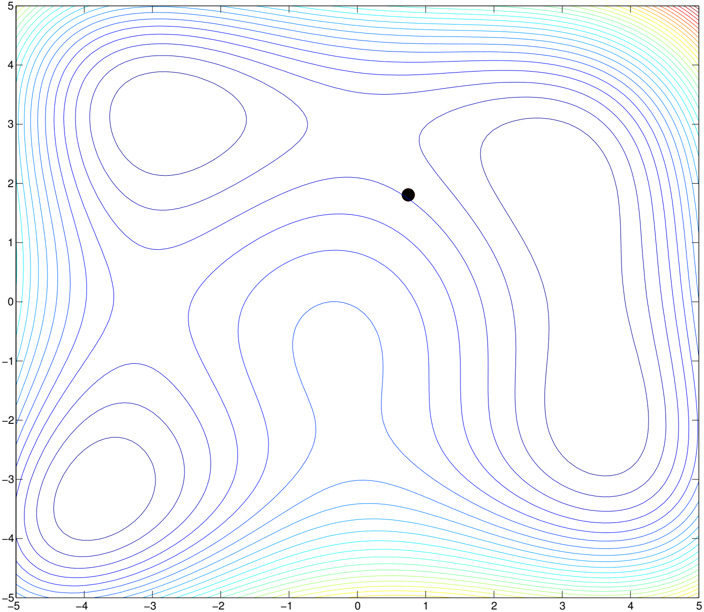
Approximate Gradients



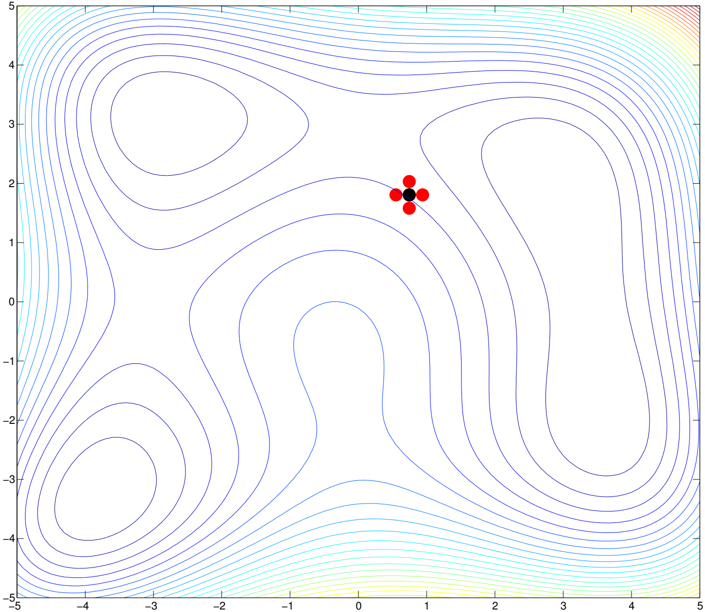
Approximate Gradients



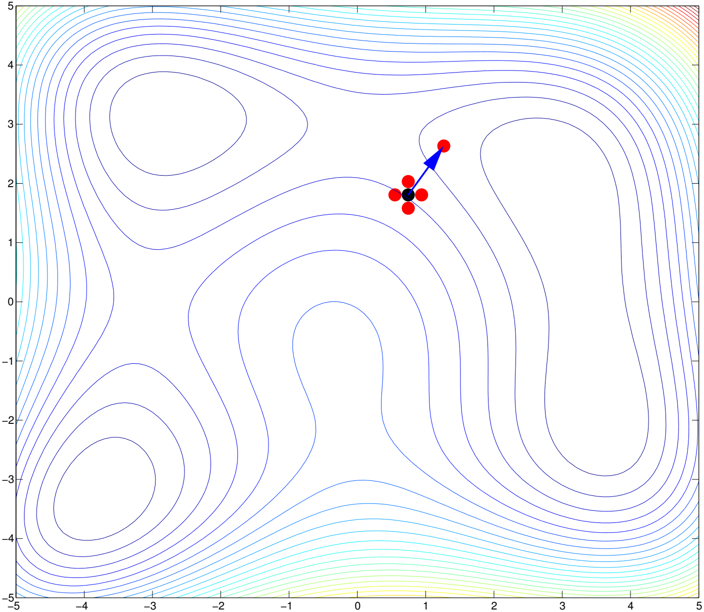
Approximate Gradients



Approximate Gradients



Approximate Gradients



Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$
  - For Kiefer-Wolfowitz,

$$G_i(x^k) = \frac{\bar{f}(x^k + c_k e_i) - \bar{f}(x^k - c_k e_i)}{2c_k}$$

where e_i is the ith column of I_n.

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$
  - For Spall's SPSA,

$$G_i(x^k) = \frac{\bar{f}(x^k + c_k \delta^k) - \bar{f}(x^k - c_k \delta^k)}{2 c_k \delta_i^k}$$

where $\delta^k \in \mathbb{R}^n$ is a random perturbation vector

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$
- a_k is a sequence of step sizes

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$
- a_k is a sequence of step sizes (specified by the user) satisfying:

$$\sum_{k=1}^{\infty} a_k = \infty \qquad \lim_{k\to\infty} a_k = 0$$

Stochastic Approximation

Iterates usually have the form:

$$x^{k+1} = x^k + a_k G(x^k),$$

where

- $G(x^k)$ is a cheap, unbiased estimate for $\nabla f(x^k)$

- a_k is a sequence of step sizes (specified by the user) satisfying:

$$\sum_{k=1}^{\infty} a_k = \infty \qquad \lim_{k\to\infty} a_k = 0$$

Algorithm performance depends significantly on sequence a_k.

Modify Existing Methods for Stochastic

Take a favorite method and repeatedly evaluate the function at points of interest.

- Stochastic approximation modified by Dupuis, Simha (1991)
- Response surface methods modified by Chang et al. (2012)
- UOBYQA modified by Deng, Ferris (2006)
- Nelder-Mead modified by Tomick et al. (1995)
- DIRECT modified by Deng, Ferris (2007)

Modify Existing Methods for Stochastic

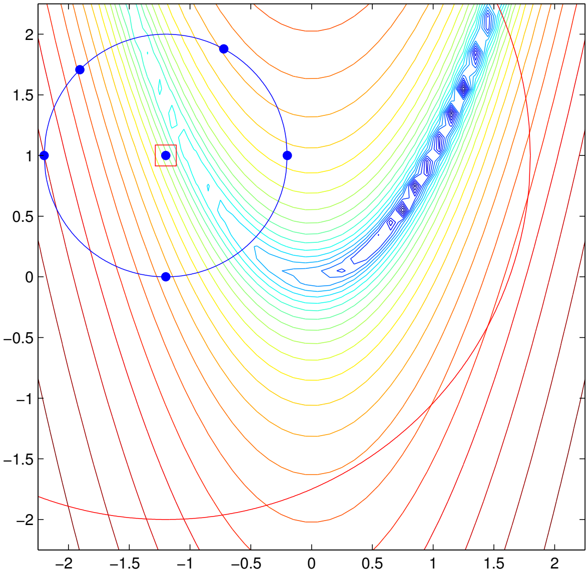
Take a favorite method and repeatedly evaluate the function at points of interest.

- Stochastic approximation modified by Dupuis, Simha (1991)
- Response surface methods modified by Chang et al. (2012)
- UOBYQA modified by Deng, Ferris (2006)
- Nelder-Mead modified by Tomick et al. (1995)
- DIRECT modified by Deng, Ferris (2007)

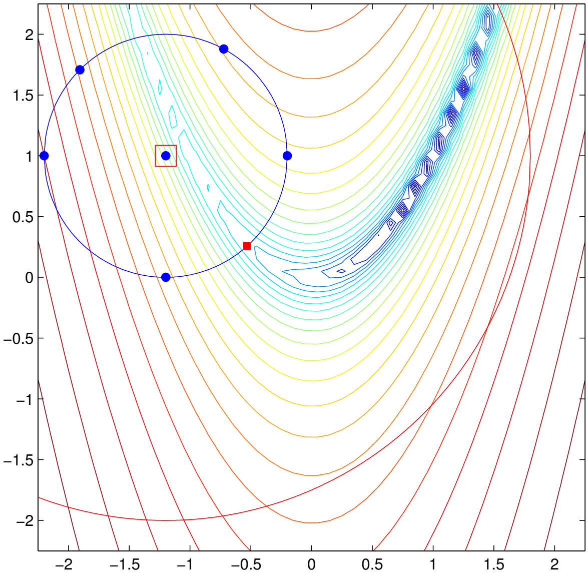
There are two downsides to such an approach:

1. Repeated sampling provides information about the noise ϵ, not f.
2. If the noise is deterministic, no information is gained.

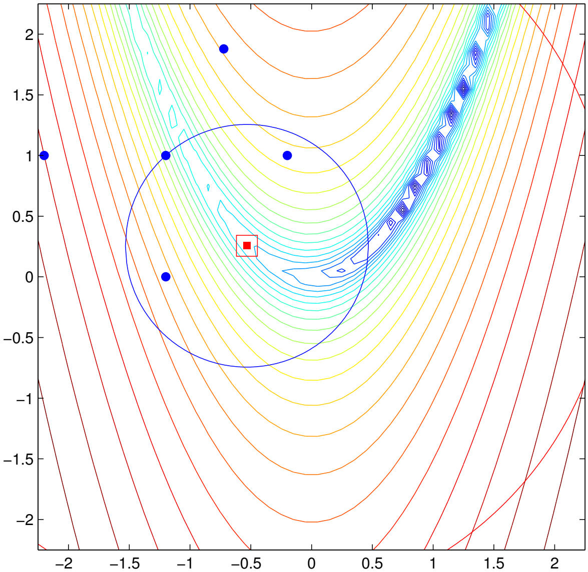
Model-based Methods



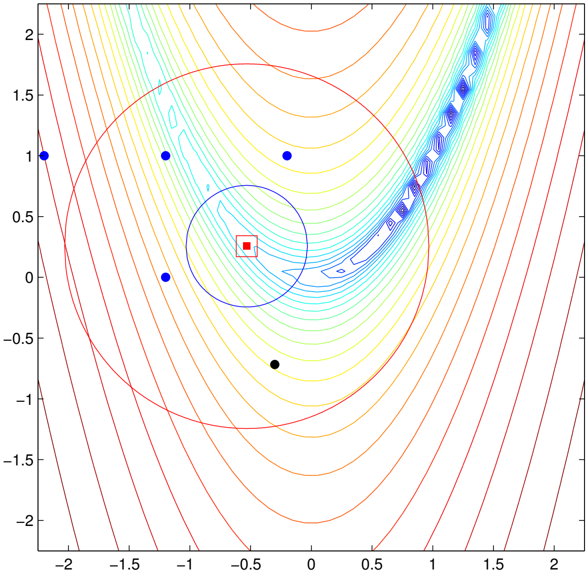
Model-based Methods



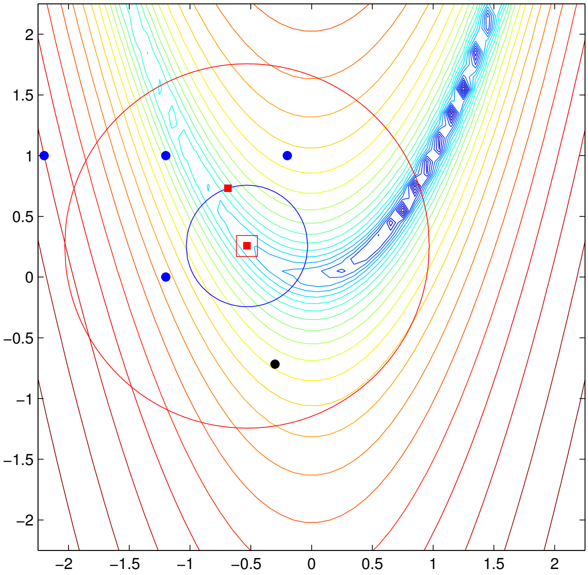
Model-based Methods



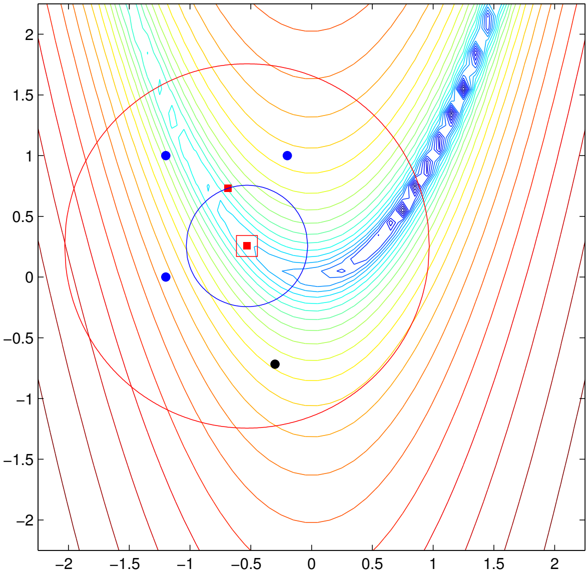
Model-based Methods



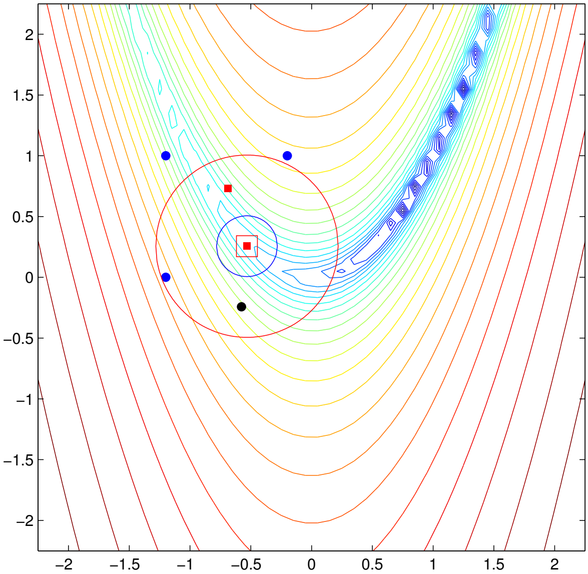
Model-based Methods



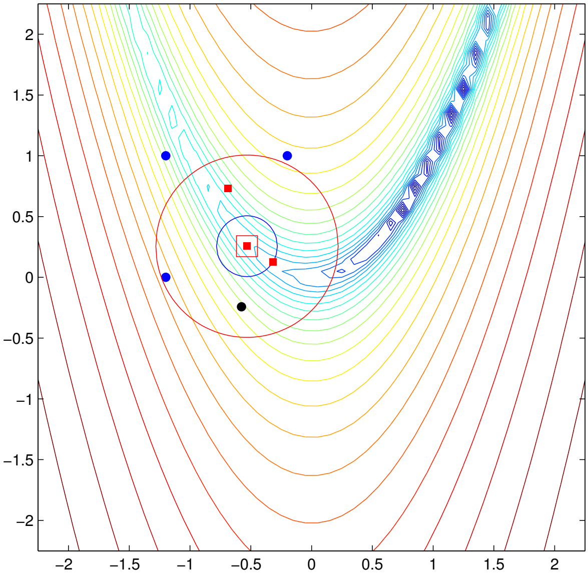
Model-based Methods



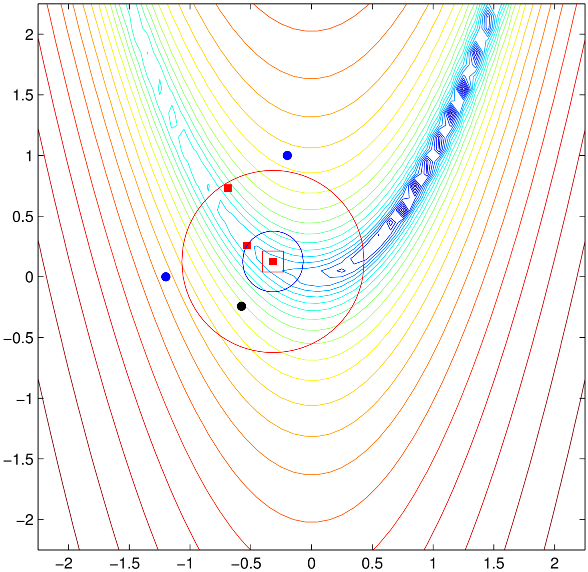
Model-based Methods



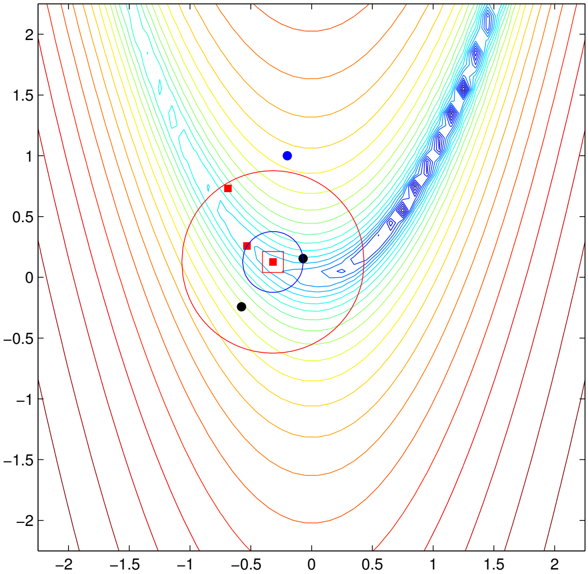
Model-based Methods



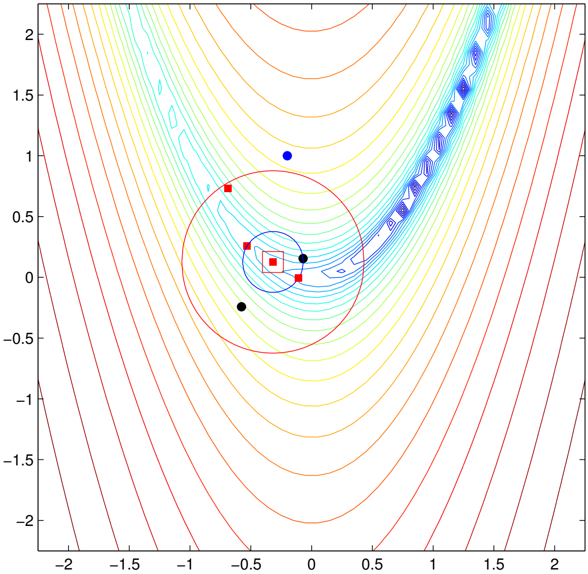
Model-based Methods



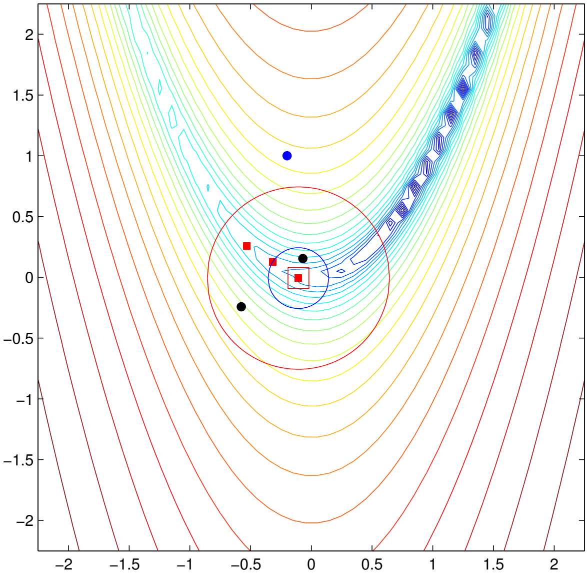
Model-based Methods



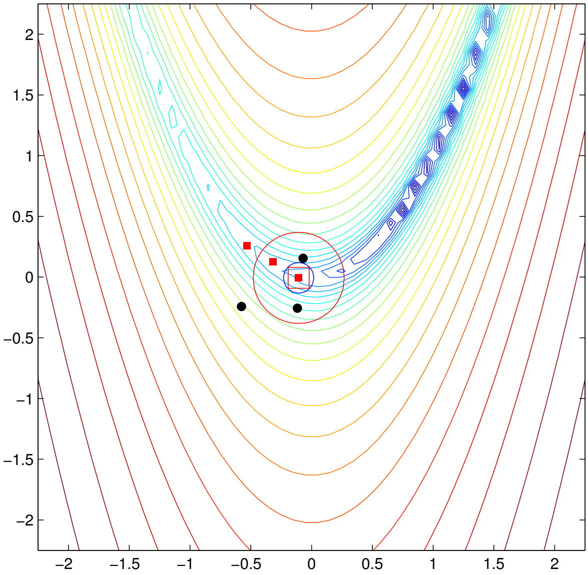
Model-based Methods



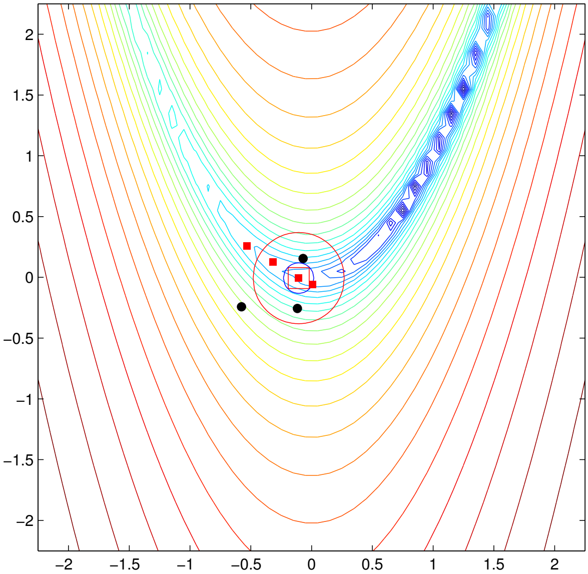
Model-based Methods



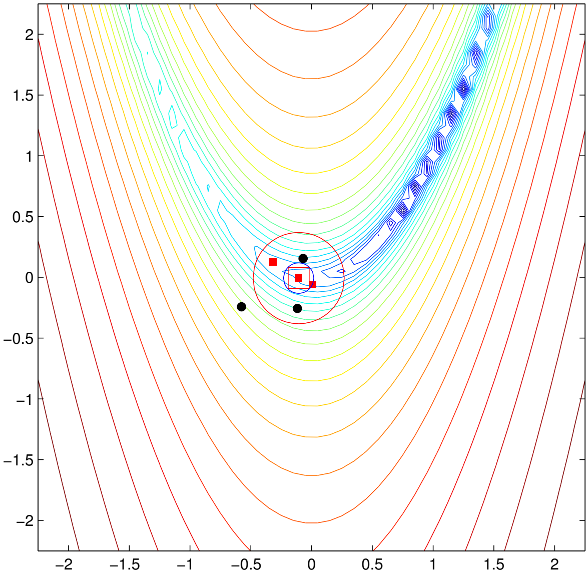
Model-based Methods



Model-based Methods



Model-based Methods



DFO warnings

- Be careful
  1) A problem can be written as scalar output, black box
  2) An algorithm exists to optimize scalar output, black box function

  1) and 2) true doesn't mean the algorithm should be used

DFO warnings

- Be careful
  1) A problem can be written as scalar output, black box
  2) An algorithm exists to optimize scalar output, black box function

1) and 2) true doesn't mean the algorithm should be used

$$\underset{x}{\text{minimize}}\, f(x) = \|Ax - b\|$$

DFO warnings

- Be careful
  1) A problem can be written as scalar output, black box
  2) An algorithm exists to optimize scalar output, black box function

  1) and 2) true doesn't mean the algorithm should be used

$$\underset{x}{\text{minimize}}\, f(x) = \|Ax - b\|$$

- If your problem has derivatives, please use them. If you don't have them...
  - Algorithmic Differentiation (AD) is wonderful

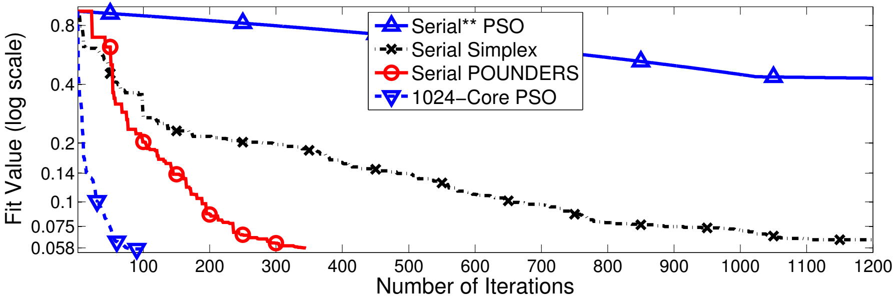
- Does the problem have structure? Avoid black boxes

Opening up the black box

$$f(x) = \sum_{i=1}^{r} (F_i(x) - T_i)^2$$

Can either have a solver that uses $f(x)$ or $[F_1(x), \ldots, F_r(x)]$.

Opening up the black box

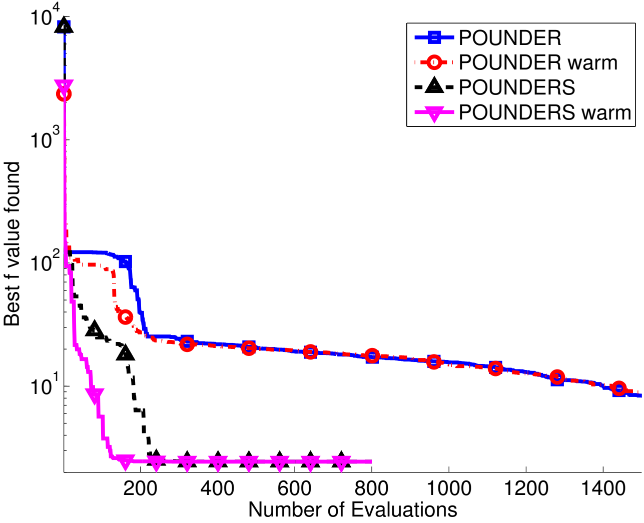


Tuning quadrapole moments for a particle accelerator simulation.

$$f(x) = \sum_{i=1}^{r} (F_i(x) - T_i)^2$$

Can either have a solver that uses $f(x)$ or $[F_1(x), \ldots, F_r(x)]$.

Opening up the black box

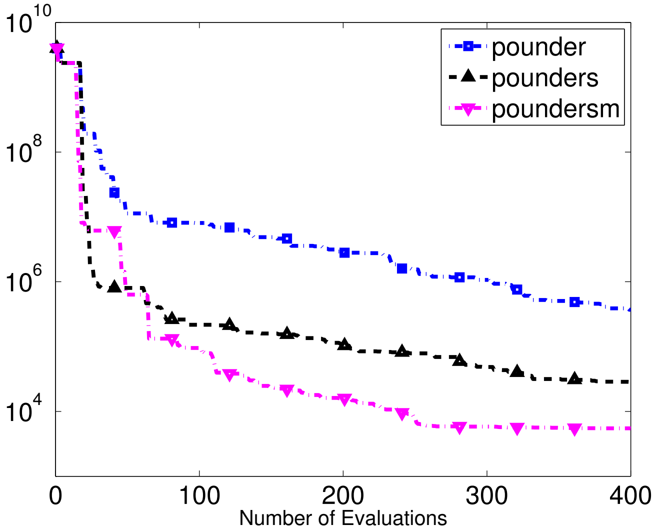


Energy density functional calibrations.

$F(x) : \mathbb{R}^{16} \to \mathbb{R}^{2049}$

$2n$ experimental design around starting point

Opening up the black box

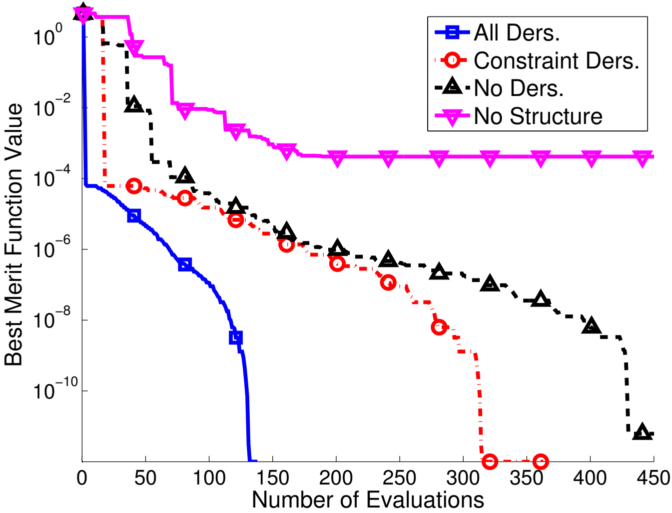


Energy density functional calibrations.

$$F(x) : \mathbb{R}^{16} \to \mathbb{R}^{2049}$$

$$F_i(x) = B([x]_{1-13}) + g(x)$$

Opening up the black box



Small gas network problem.

- 15 variables
- 11 constraints

- $\nabla_x f$ and $\nabla_x c$
- f and $\nabla_x c$
- f and c (separate) black boxes
- Penalizing constraints

Exploiting Structure

- Nonsmooth, composite optimization

$$\underset{x}{\text{minimize}}\, f(x) = h(F(x))$$

where ∇F is unavailable but ∂h is known

Exploiting Structure

- Nonsmooth, composite optimization

$$\underset{x}{\text{minimize}}\, f(x) = h(F(x))$$

where ∇F is unavailable but ∂h is known

- Multiple objectives

Exploiting Structure

- Nonsmooth, composite optimization

$$\underset{x}{\text{minimize}}\, f(x) = h(F(x))$$

where ∇F is unavailable but ∂h is known

- Multiple objectives

- Controllable accuracy

Exploiting Structure

- Nonsmooth, composite optimization

$$\underset{x}{\text{minimize}}\, f(x) = h(F(x))$$

where ∇F is unavailable but ∂h is known

- Multiple objectives

- Controllable accuracy

- Multiple local minima

Motivation

- We want to identify distinct, "high-quality", local minimizers of

$$\begin{aligned} \text{minimize } & f(x) \\ l \leq x & \leq u \\ x & \in \mathbb{R}^n \end{aligned}$$

- High-quality can be measured by more than the objective.

Motivation

- We want to identify distinct, "high-quality", local minimizers of

$$\begin{aligned} \text{minimize } f(x) \\ l \leq x \leq u \\ x \in \mathbb{R}^n \end{aligned}$$

- High-quality can be measured by more than the objective.

- Derivatives of f may or may not be available.

Motivation

- We want to identify distinct, "high-quality", local minimizers of

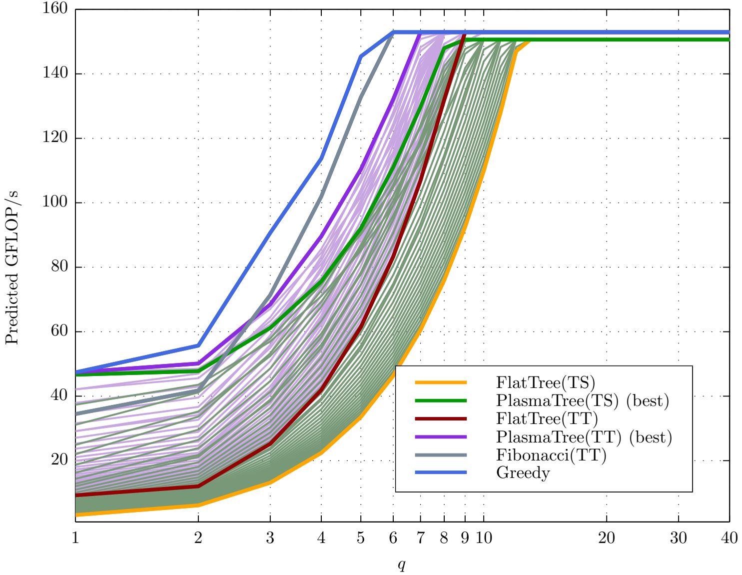
$$\begin{aligned} \text{minimize}\, f(x) \\ l \leq x \leq u \\ x \in \mathbb{R}^n \end{aligned}$$

- High-quality can be measured by more than the objective.

- Derivatives of f may or may not be available.

- The simulation f is likely using parallel resources, but it does not utilize the entire machine.

Why concurrency? Tiled QR example



[Bouwmeester, et al., Tiled QR Factorization Algorithms, 2011]

Global DFO

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

Global DFO

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$

Global DFO

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$

- Or possibly wait a long time (or forever)

Global DFO

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$

- Or possibly wait a long time (or forever)

The theory can be more than merely checking that a method generates iterates which are dense in the domain.

Global DFO

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$

- Or possibly wait a long time (or forever)

The theory can be more than merely checking that a method generates iterates which are dense in the domain.

An algorithm must trade-off between “refinement” and “exploration”.

Global DFO

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

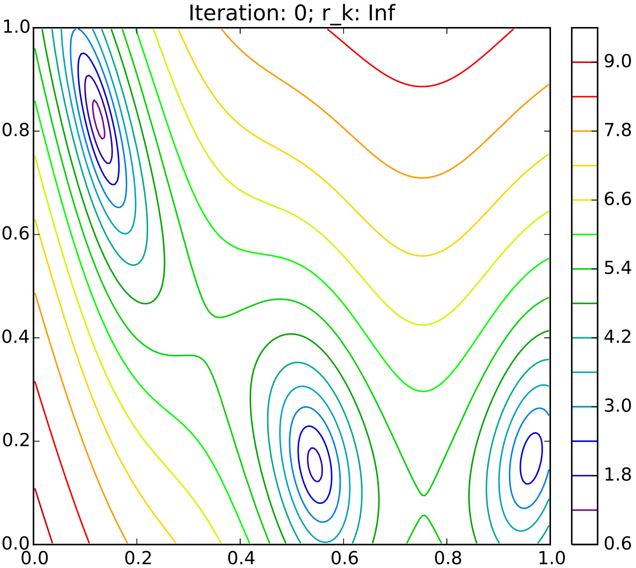
An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$
  - concurrent evaluations of f
- Or possibly wait a long time (or forever)

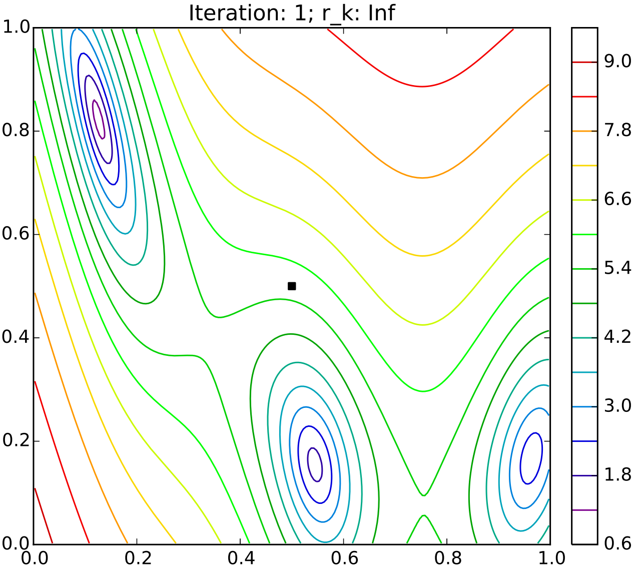
The theory can be more than merely checking that a method generates iterates which are dense in the domain.

An algorithm must trade-off between "refinement" and "exploration".

DIRECT

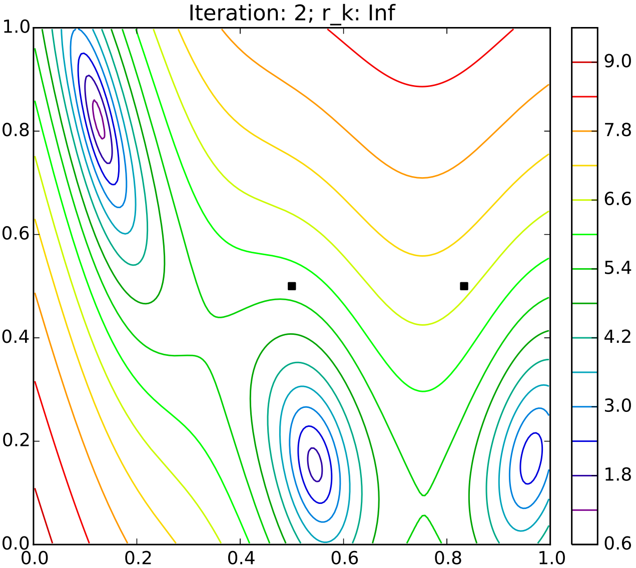

Iteration: 0; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

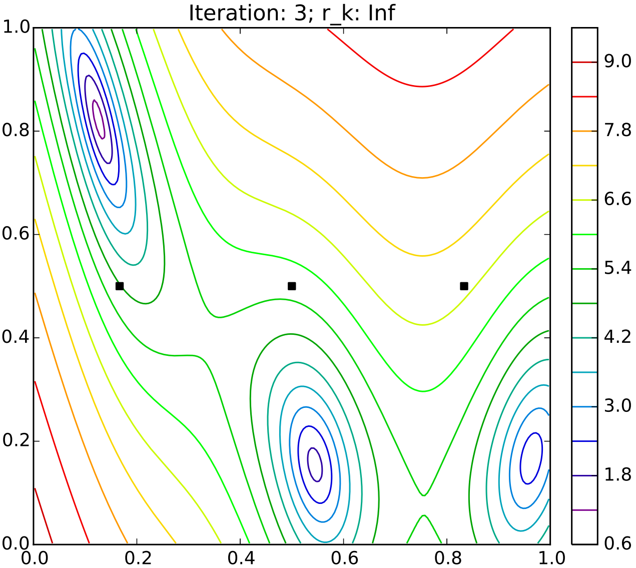


Iteration: 1; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

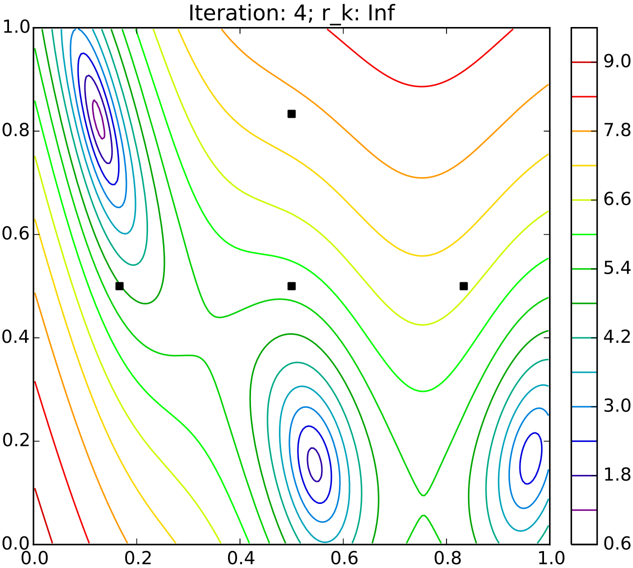
DIRECT


Iteration: 2; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

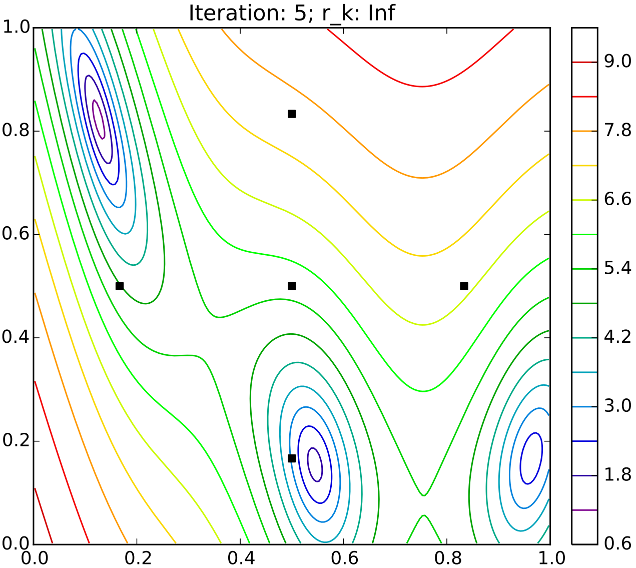

Iteration: 3; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



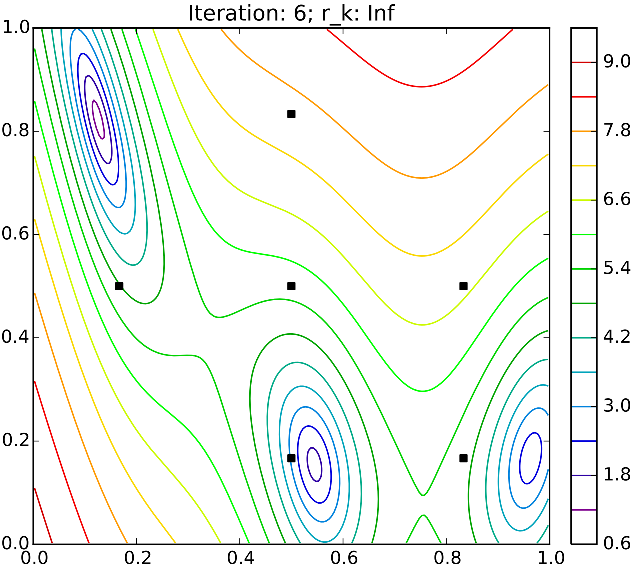
Iteration: 4; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

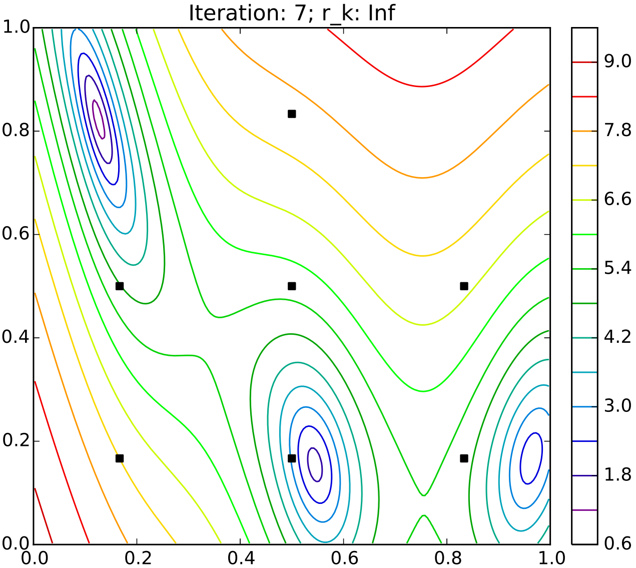


Iteration: 5; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

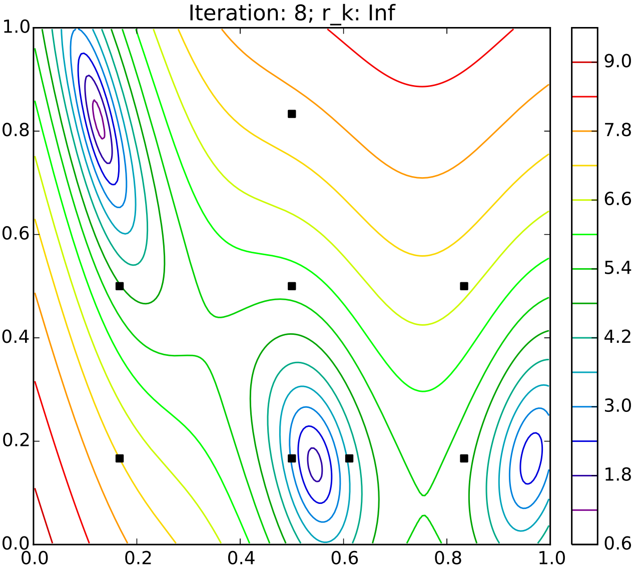

Iteration: 6; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



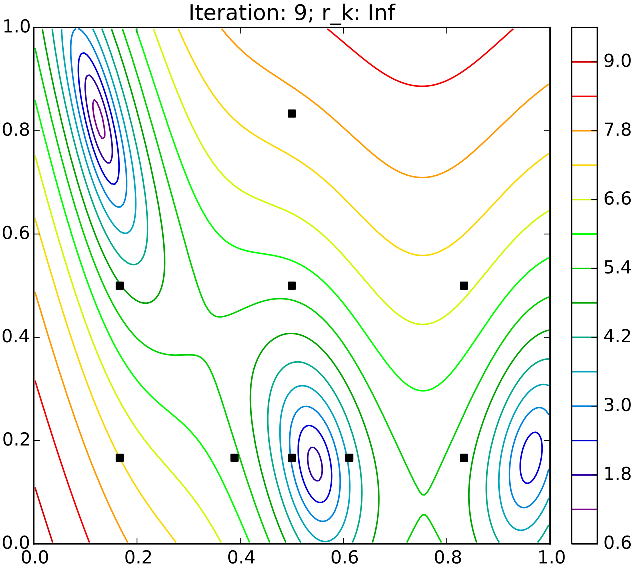
Iteration: 7; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



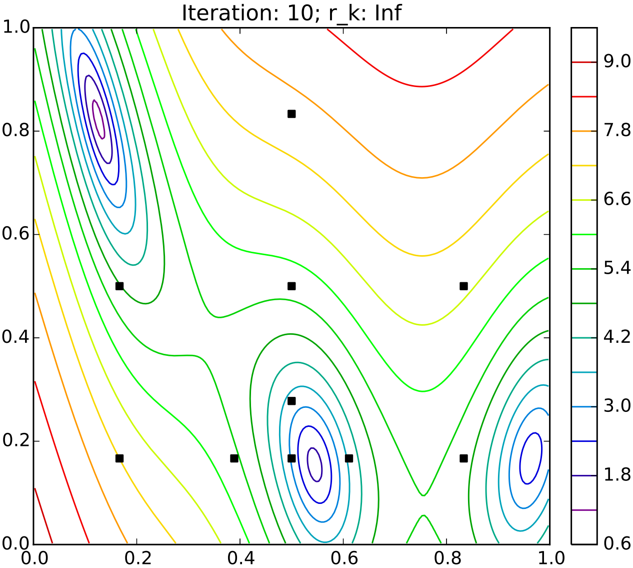
Iteration: 8; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



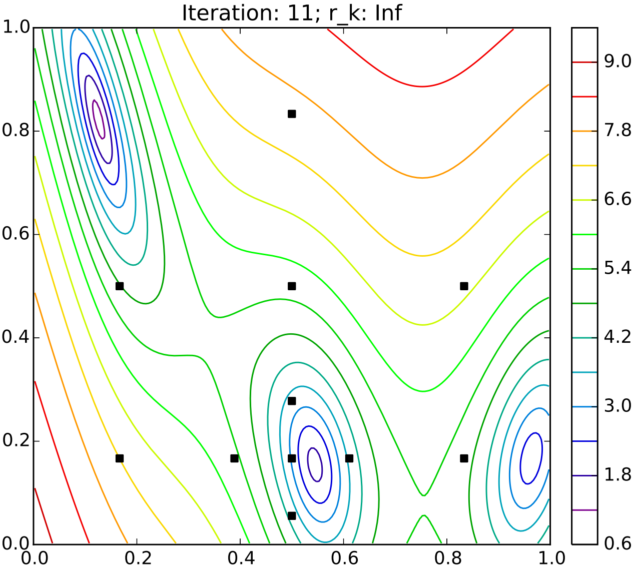
Iteration: 9; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



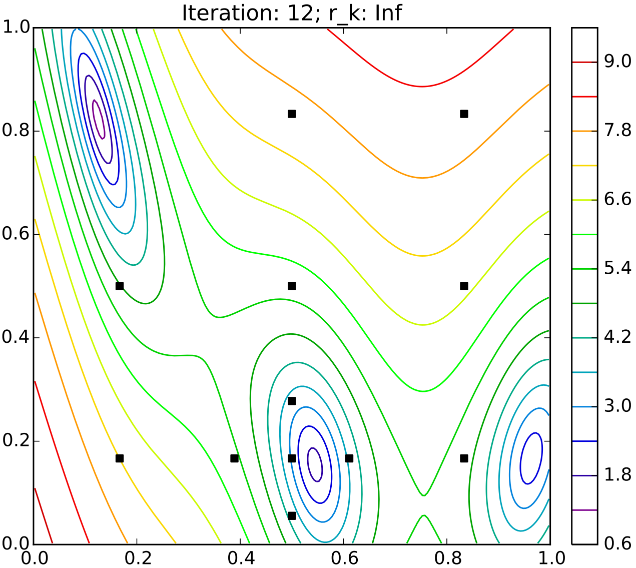
Iteration: 10; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



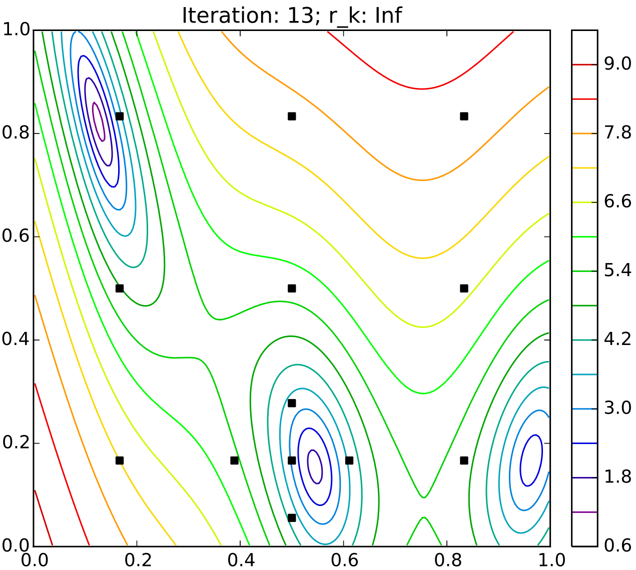
Iteration: 11; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

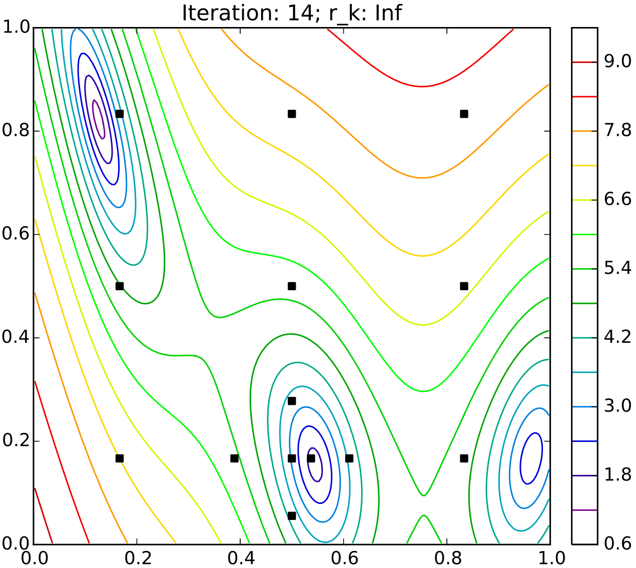


Iteration: 12; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

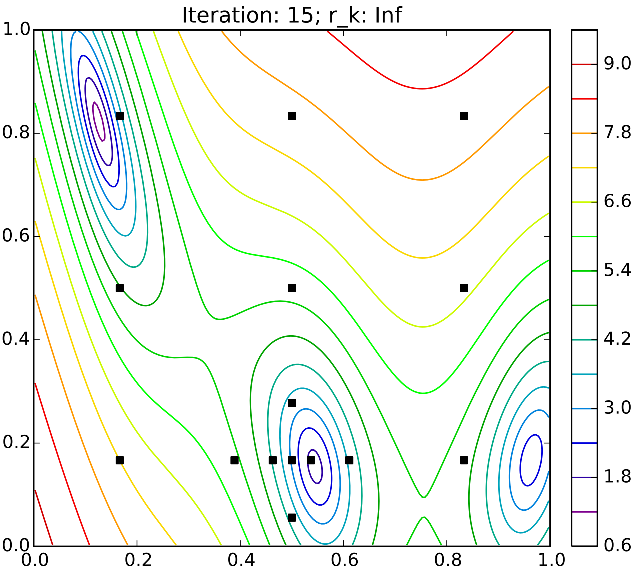
DIRECT


Iteration: 13; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

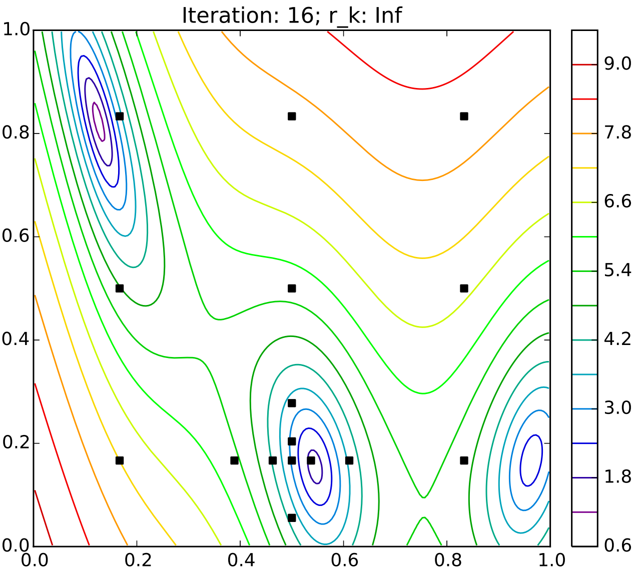

Iteration: 14; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



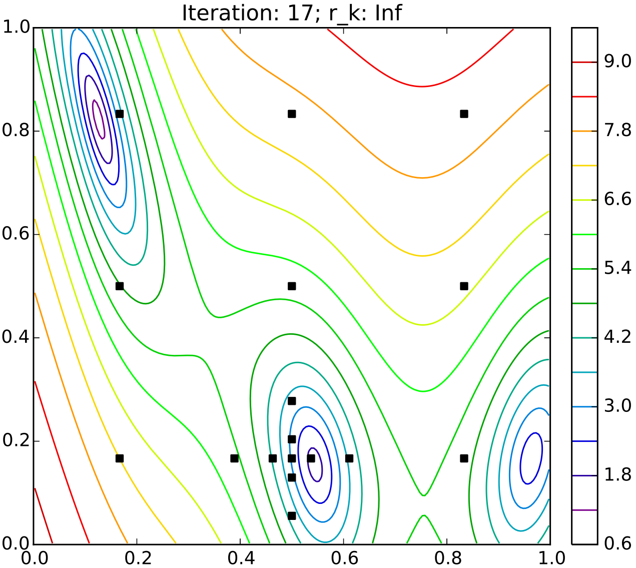
Iteration: 15; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

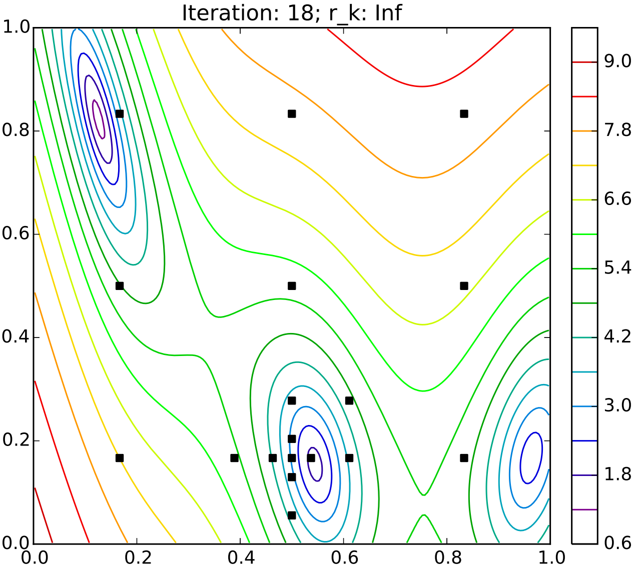


Iteration: 16; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

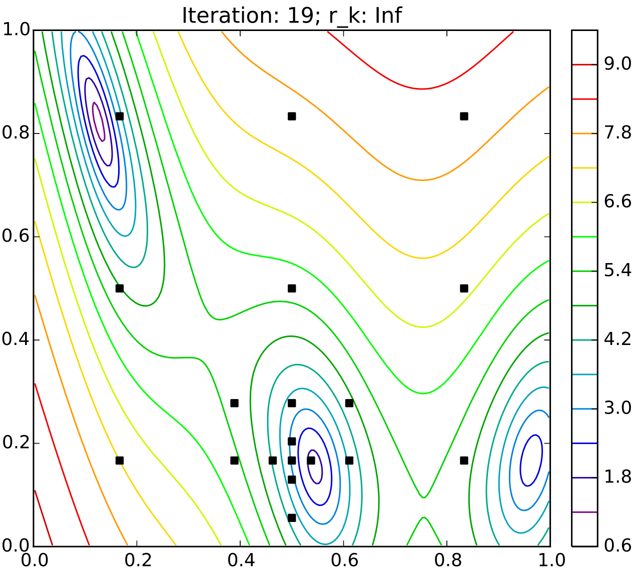

Iteration: 17; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



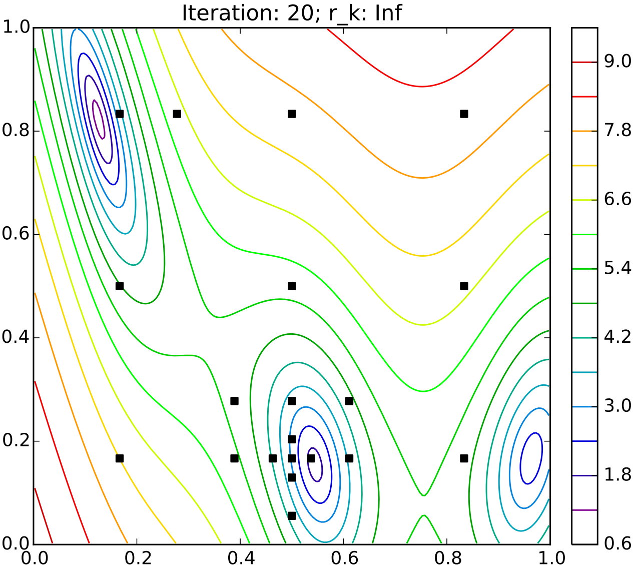
Iteration: 18; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



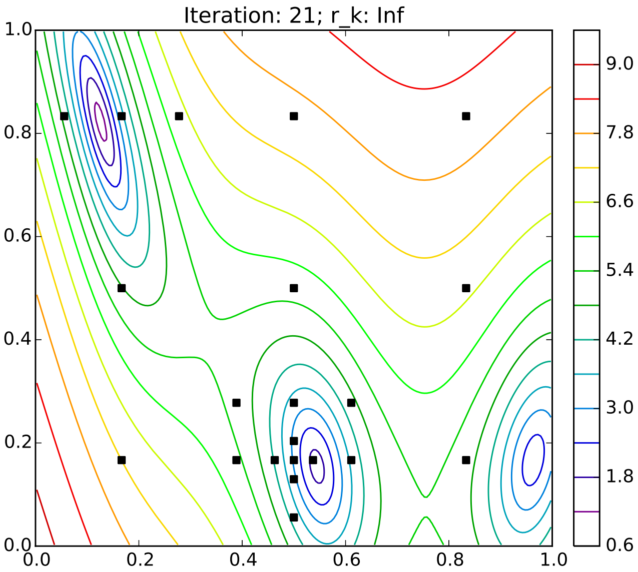
Iteration: 19; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

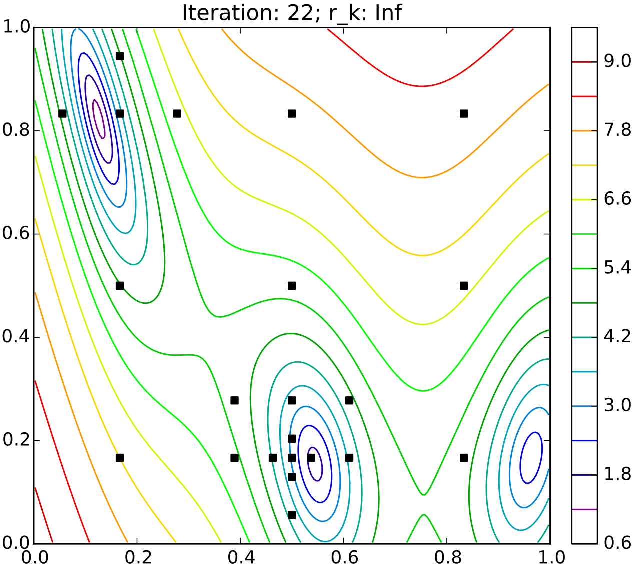


Iteration: 20; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

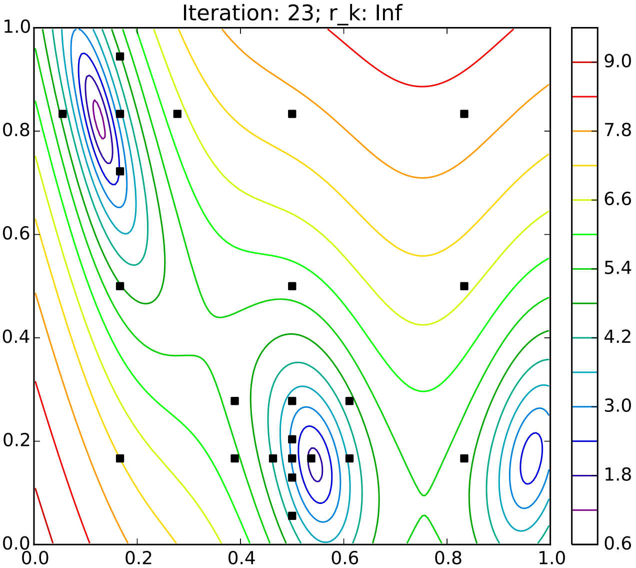

Iteration: 21; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



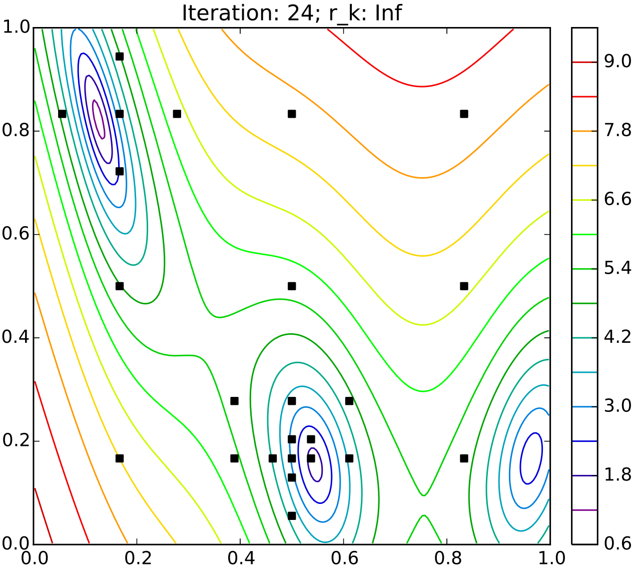
Iteration: 22; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

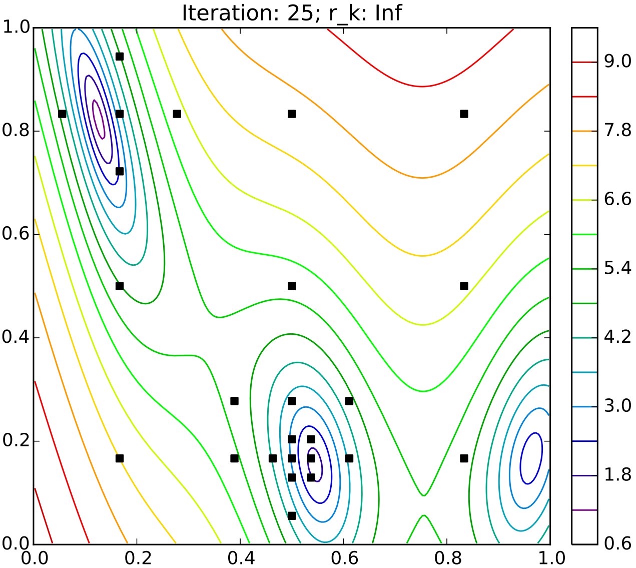


Iteration: 23; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT

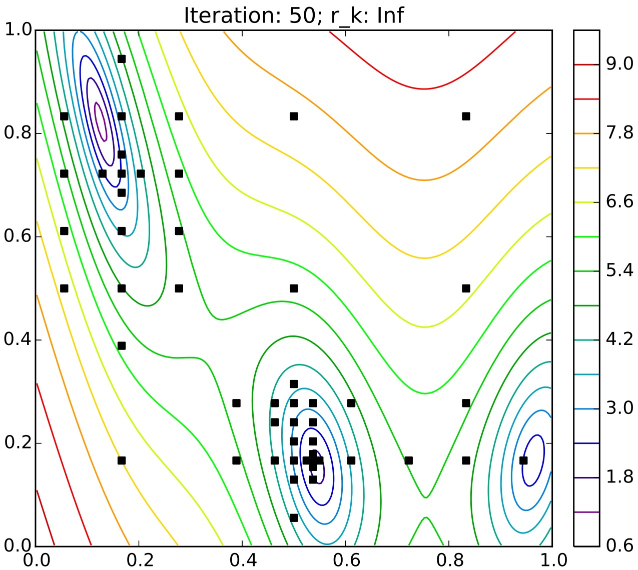

Iteration: 24; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



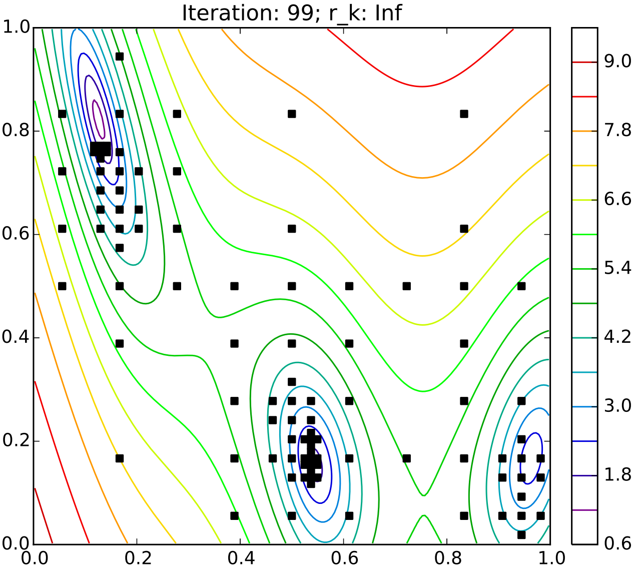
Iteration: 25; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



Iteration: 50; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

DIRECT



Iteration: 99; r_k: Inf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
9.0
7.8
6.6
5.4
4.2
3.0
1.8
0.6

Multistart Methods

- Explore by random sampling from the domain $\mathcal{D}$
- Refine by using a local optimization run from some subset of points

Multistart Methods

- Explore by random sampling from the domain $\mathcal{D}$
- Refine by using a local optimization run from some subset of points

Desire to find all minima but start only one run for each minimum

Multistart Methods

- Explore by random sampling from the domain $\mathcal{D}$
- Refine by using a local optimization run from some subset of points

Desire to find all minima but start only one run for each minimum

+ Get to use (more developed) local optimization routines.
  - least-squares objectives, nonsmooth objectives, (un)relaxable constraints, and more
+ Increased opportunity for parallelism
  - objective, local solver, and global solver

\- Can require many sequential evaluations for the local solver

Multi-Level Single Linkage

Given some local optimization routine $\mathcal{L}$:

Algorithm 1: MLSL

for $k = 1, 2, \ldots$ **do**

Sample f at N random points drawn uniformly from $\mathcal{D}$

Start $\mathcal{L}$ at all sample points x:

- that has yet to start a run
- $\nexists x_i : \|x - x_i\| \leq r_k$ and $f(x_i) < f(x)$

[Rinnooy Kan and Timmer, *Mathematical Programming*, 39(1):57–78, 1987]

Multi-Level Single Linkage

Given some local optimization routine $\mathcal{L}$:

Algorithm 1: MLSL

for $k = 1, 2, \ldots$ **do**

Sample f at N random points drawn uniformly from $\mathcal{D}$

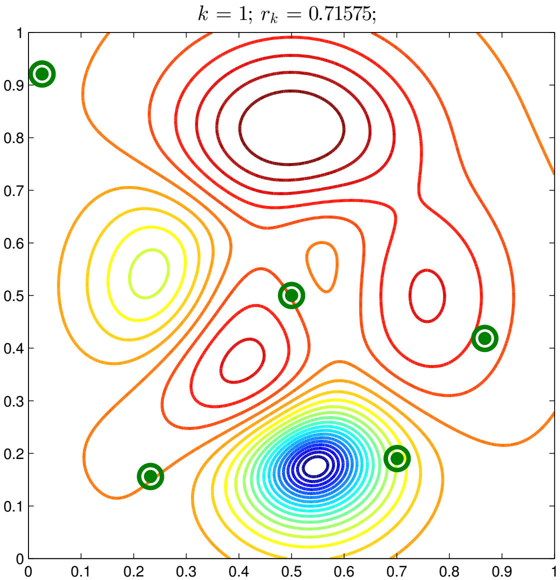
Start $\mathcal{L}$ at all sample points x:

- that has yet to start a run
- $\nexists x_i : \|x - x_i\| \leq r_k$ and $f(x_i) < f(x)$

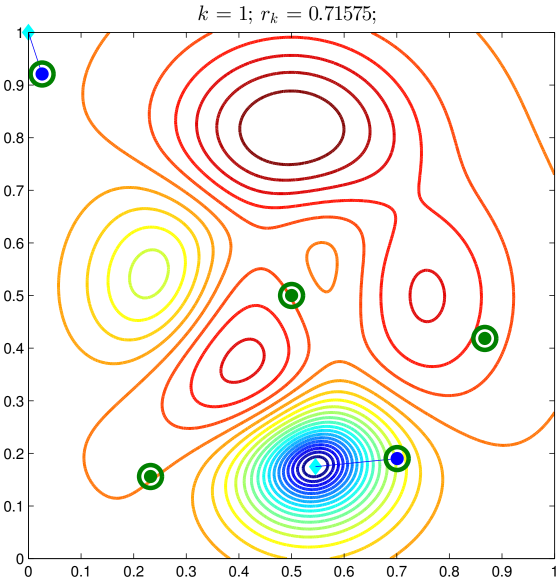
[Rinnooy Kan and Timmer, *Mathematical Programming*, 39(1):57–78, 1987]

- Doesn't naturally translate when evaluations of f are limited
- Ignores some points when deciding where to start $\mathcal{L}$

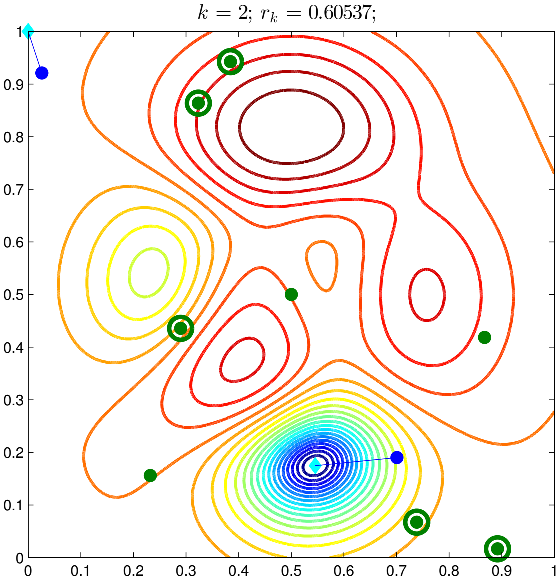
Multi-Level Single Linkage



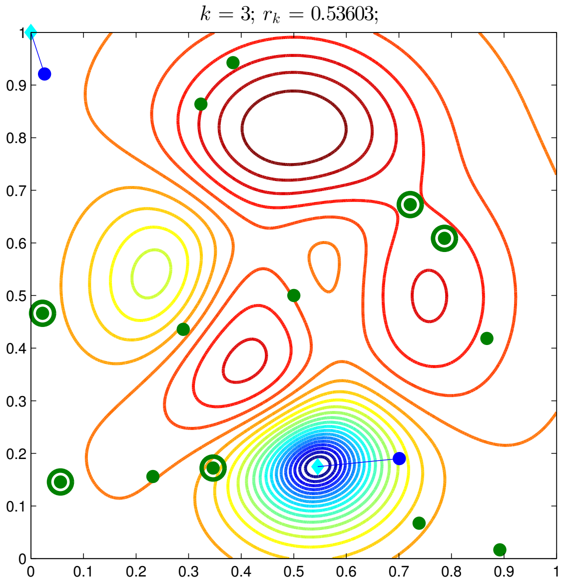
Multi-Level Single Linkage



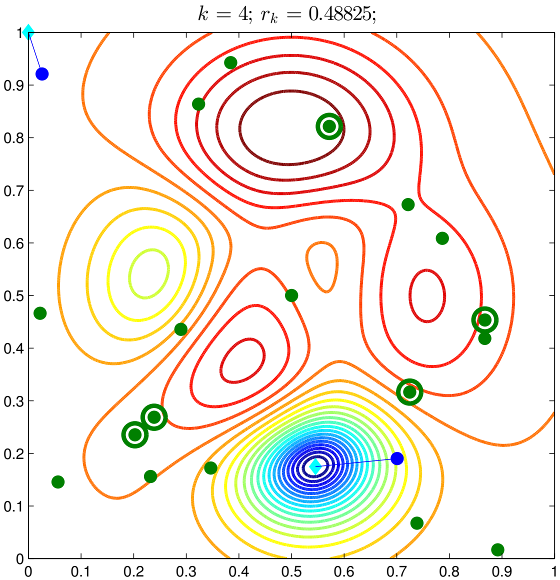
Multi-Level Single Linkage



Multi-Level Single Linkage

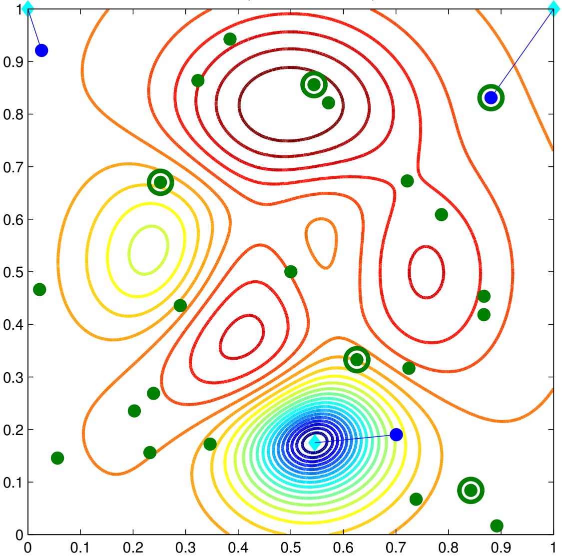


Multi-Level Single Linkage

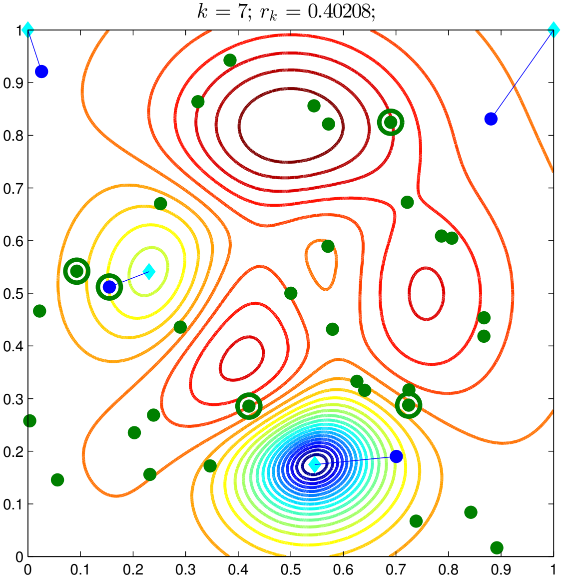


Multi-Level Single Linkage

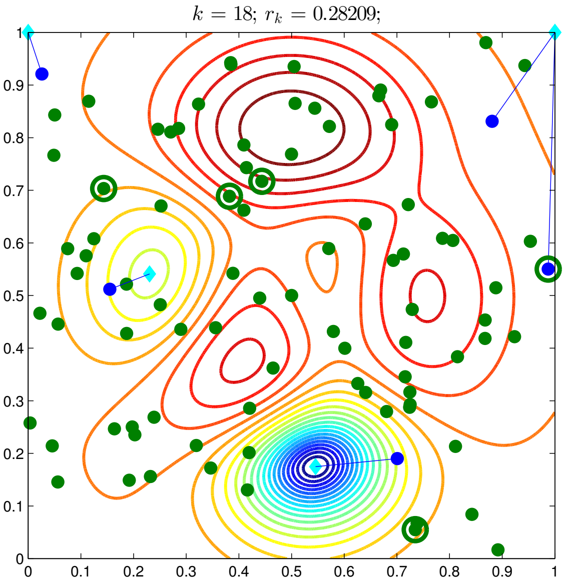
$k = 5;\ r_k = 0.45268;$



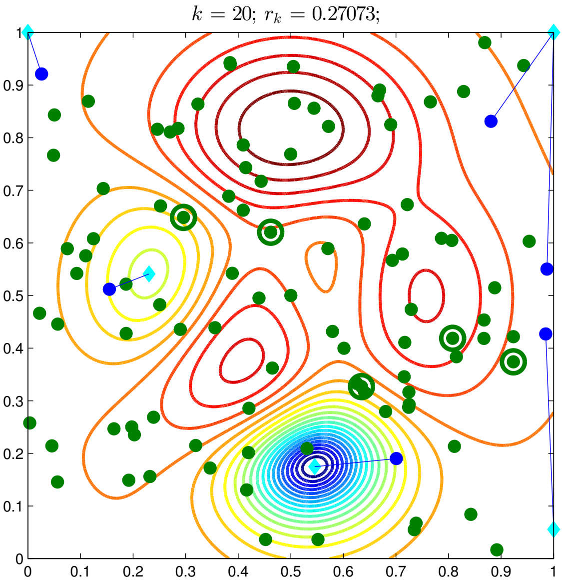
Multi-Level Single Linkage



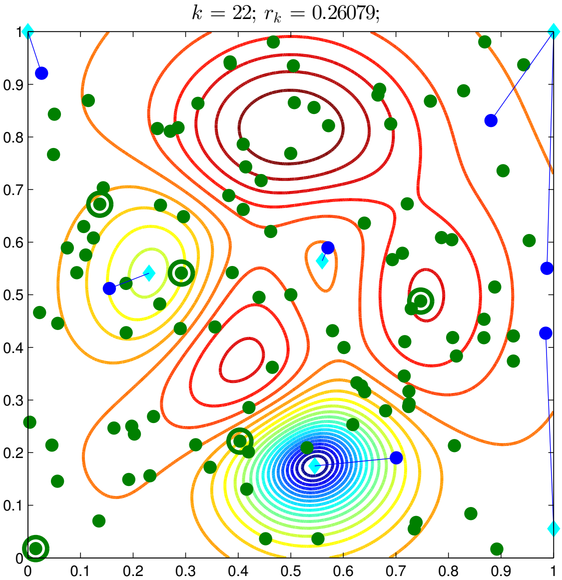
Multi-Level Single Linkage



Multi-Level Single Linkage



Multi-Level Single Linkage



Multi-Level Single Linkage

- $f \in C^2$, with local minima in the interior of $\mathcal{D}$, and the distance between these minima is bounded away from zero.
- $\mathcal{L}$ is strictly descent and converges to a minimum (not a stationary point).
- $$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\Gamma\left(1 + \frac{n}{2}\right) \mathrm{vol}\left(\mathcal{D}\right) \frac{\sigma \log kN}{kN}} \qquad (1)$$

Theorem

If $r_k \to 0$, all local minima will be found almost surely.

Multi-Level Single Linkage

- $f \in C^2$, with local minima in the interior of $\mathcal{D}$, and the distance between these minima is bounded away from zero.
- $\mathcal{L}$ is strictly descent and converges to a minimum (not a stationary point).
- $$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\Gamma\left(1 + \frac{n}{2}\right) \mathrm{vol}\left(\mathcal{D}\right) \frac{\sigma \log kN}{kN}} \tag{1}$$

Theorem

If $r_k \to 0$, all local minima will be found almost surely.

If r_k is defined by (1) with $\sigma > 4$, even if the sampling continues forever, the total number of local searches started is finite almost surely.

BAMLM

MLSL: (S2)–(S4)

$\hat{x} \in \mathcal{S}_k$

(S2) $\nexists x \in \mathcal{S}_k$ *with* $[\|\hat{x} - x\| \leq r_k$ *and* $f(x) < f(\hat{x})]$

(S3) $\hat{x}$ *has not started a local optimization run*

(S4) $\hat{x}$ *is at least* μ *from* $\partial\mathcal{D}$ *and* ν *from known local minima*

BAMLM

MLSL: (S2)–(S4)

$\hat{x} \in \mathcal{S}_k$

(S1) $\nexists x \in \mathcal{L}_k$ with $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

(S2) *$\nexists x \in \mathcal{S}_k$ with $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$*

(S3) *$\hat{x}$ has not started a local optimization run*

(S4) *$\hat{x}$ is at least μ from $\partial\mathcal{D}$ and ν from known local minima*

BAMLM: (S1)–(S4), (L1)–(L6)

$\hat{x} \in \mathcal{L}_k$

(L1) $\nexists x \in \mathcal{L}_k$ $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

(L2) $\nexists x \in \mathcal{S}_k$ with $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

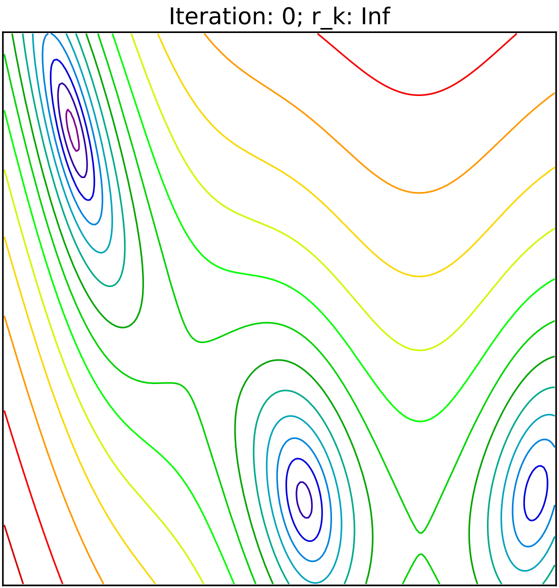
(L3) $\hat{x}$ has not started a local optimization run

(L4) $\hat{x}$ is at least μ from $\partial\mathcal{D}$ and ν from known local minima

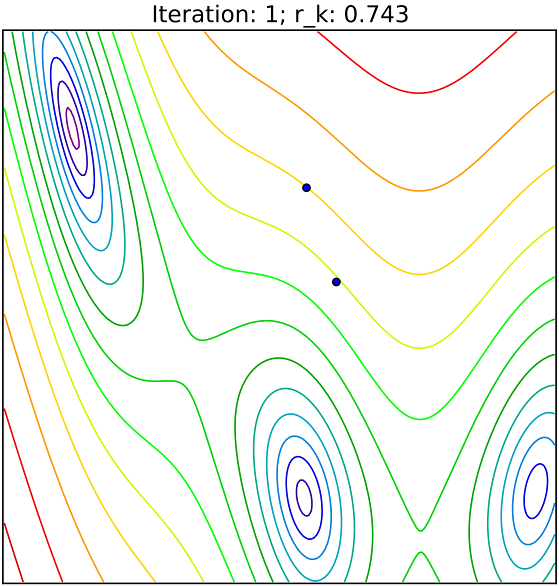
(L5) $\hat{x}$ is not in an active local optimization run and has not been ruled stationary

(L6) $\exists r_k$-descent path in $\mathcal{H}_k$ from some $x \in \mathcal{S}_k$ satisfying (S2-S4) to $\hat{x}$

BAMLM

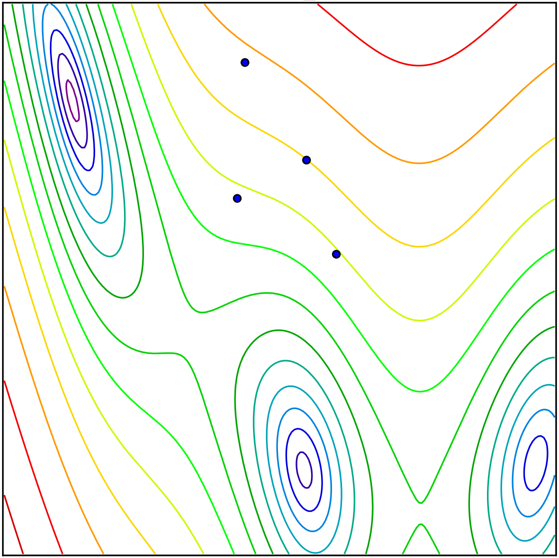


BAMLM



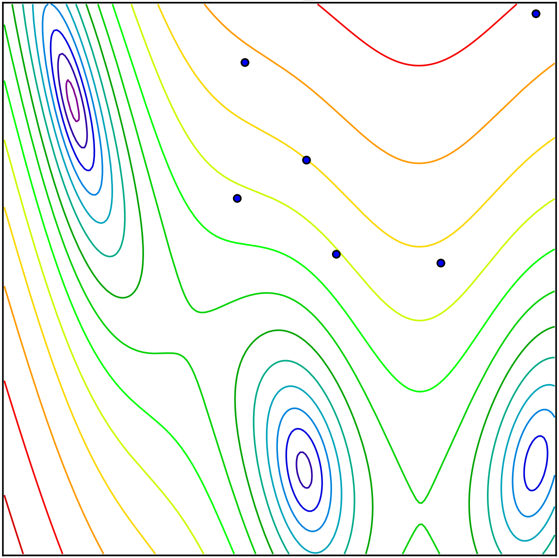
BAMLM

Iteration: 2; r_k: 0.743

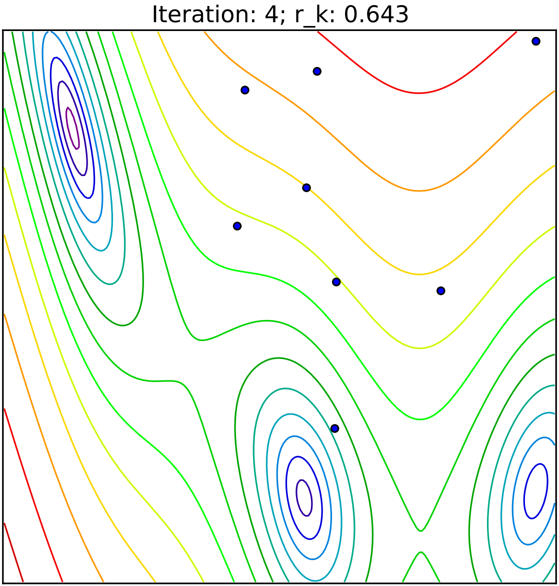


BAMLM

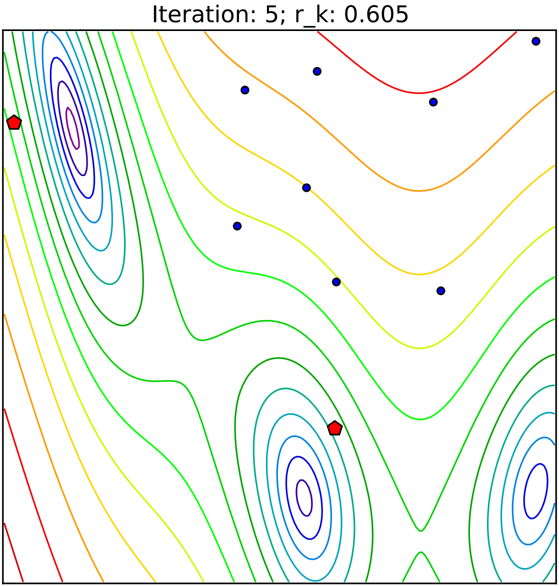
Iteration: 3; r_k: 0.689



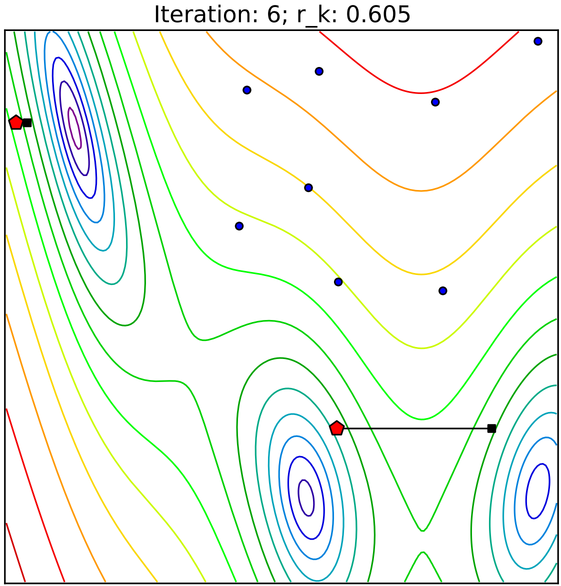
BAMLM



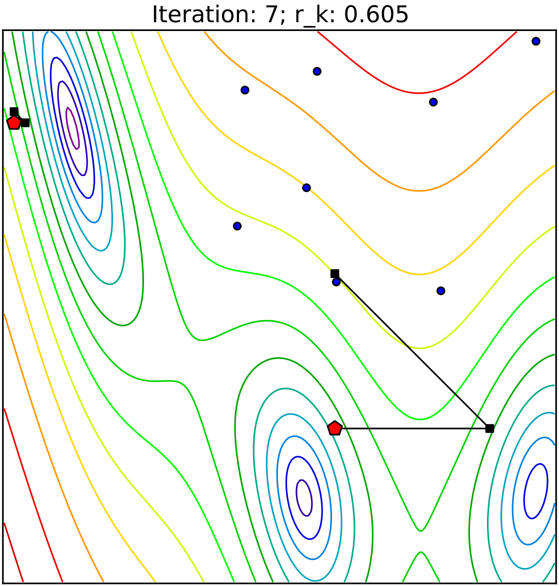
BAMLM



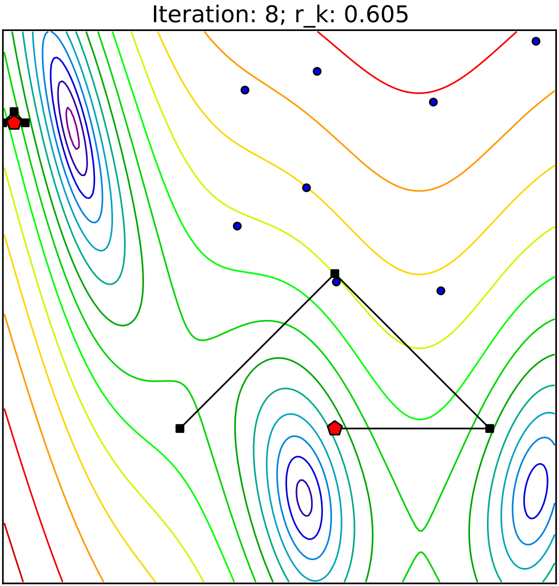
BAMLM



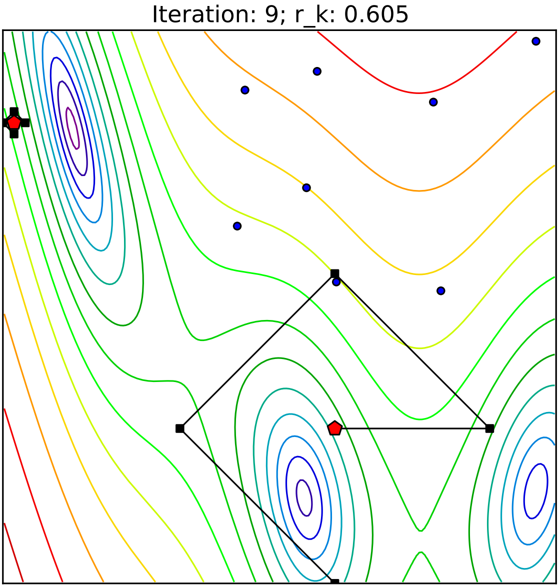
BAMLM



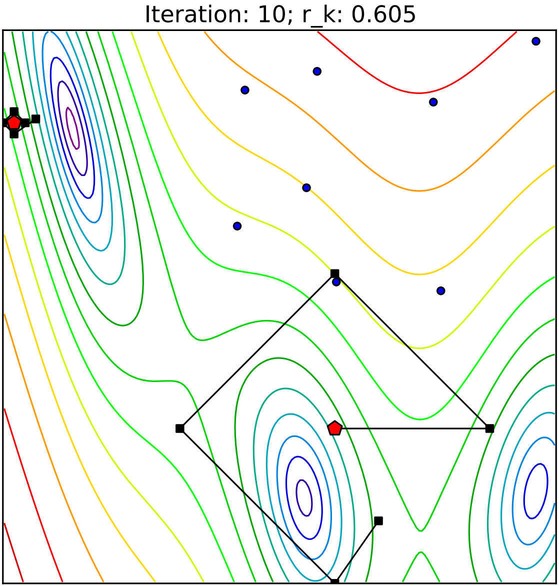
BAMLM



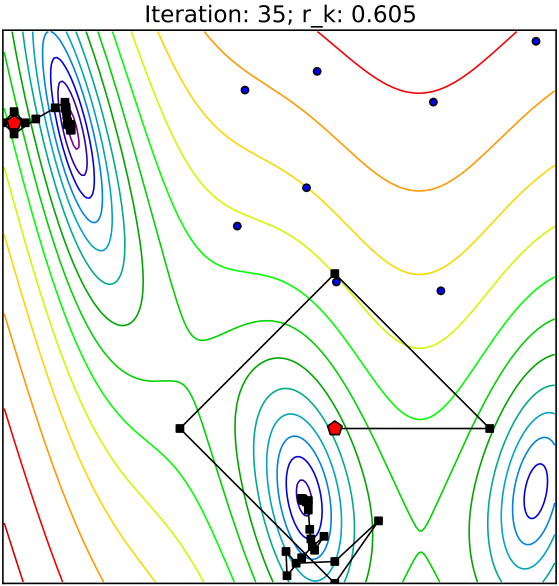
BAMLM



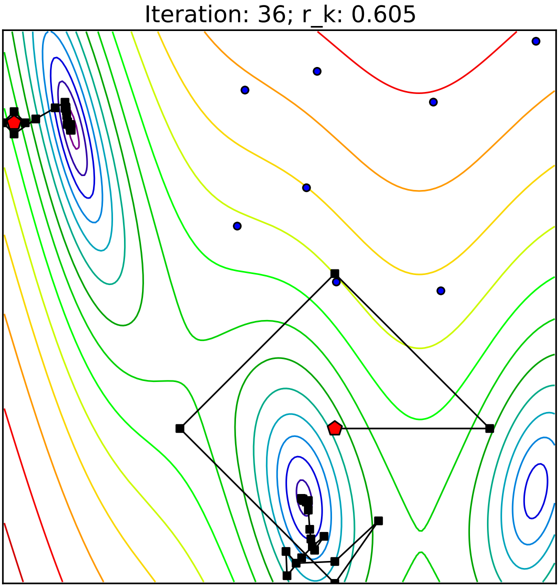
BAMLM



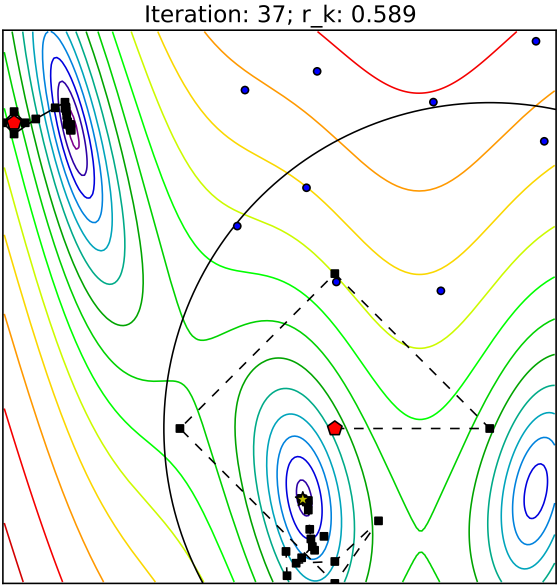
BAMLM



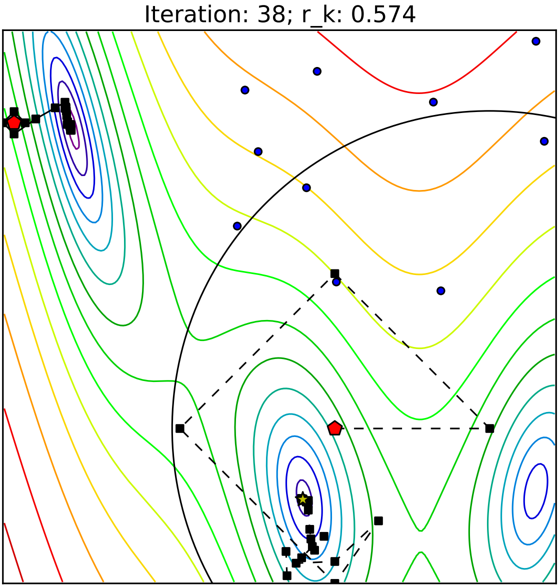
BAMLM



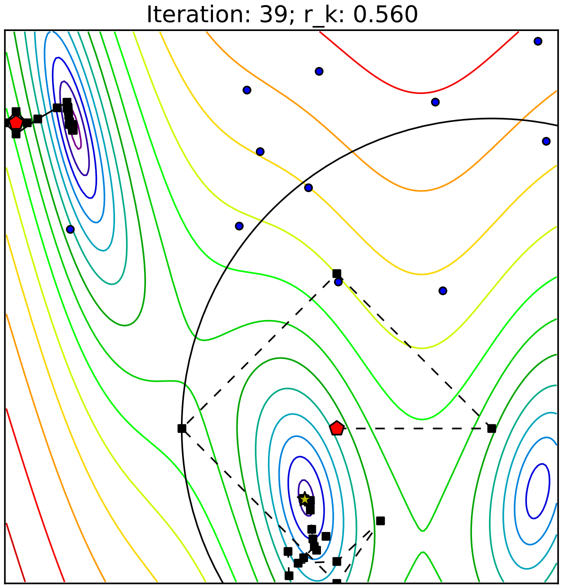
BAMLM



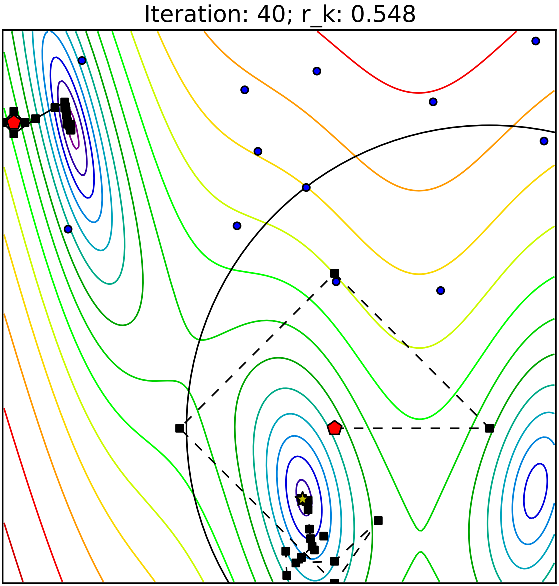
BAMLM



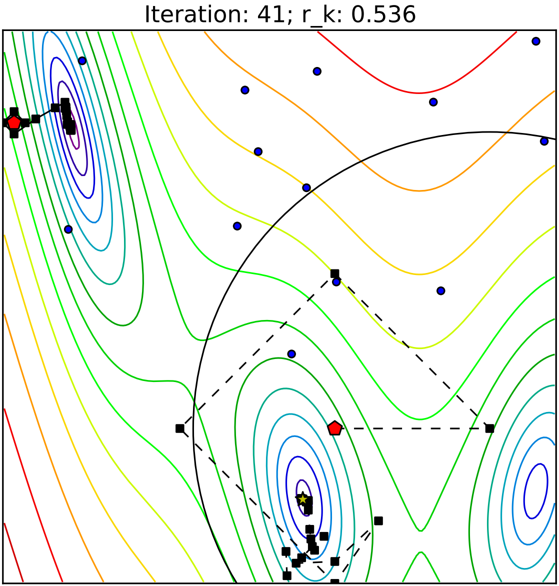
BAMLM



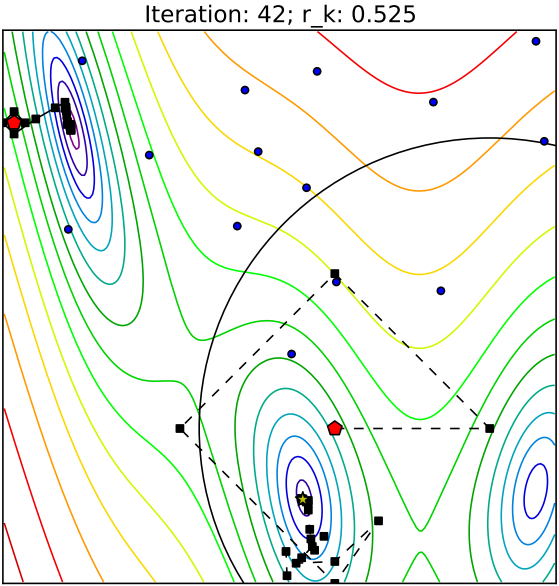
BAMLM



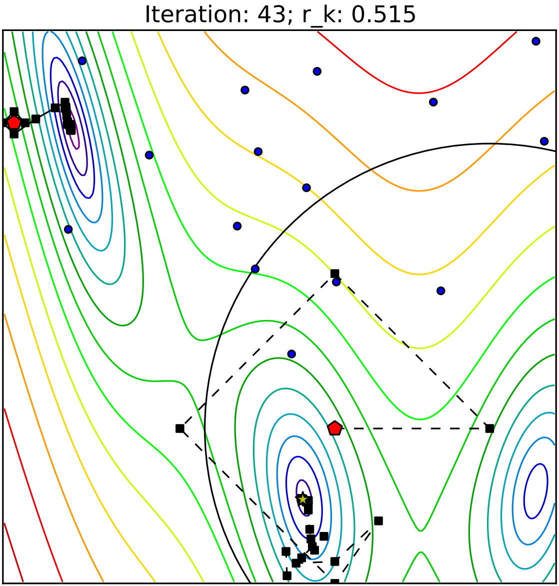
BAMLM



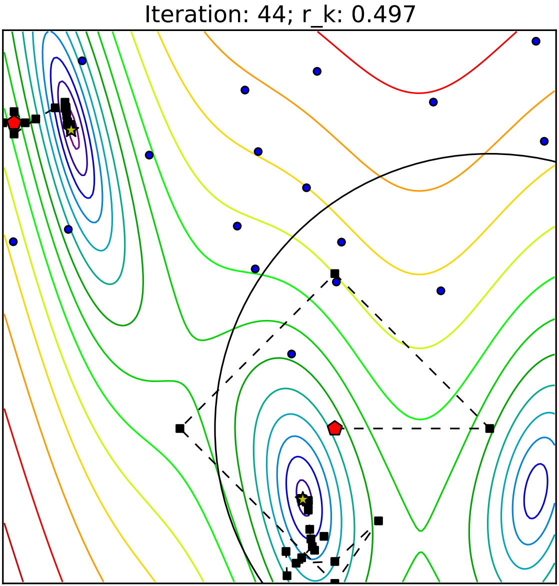
BAMLM



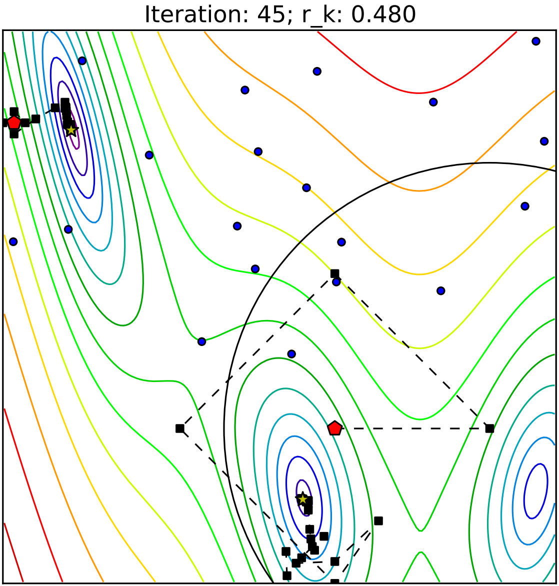
BAMLM



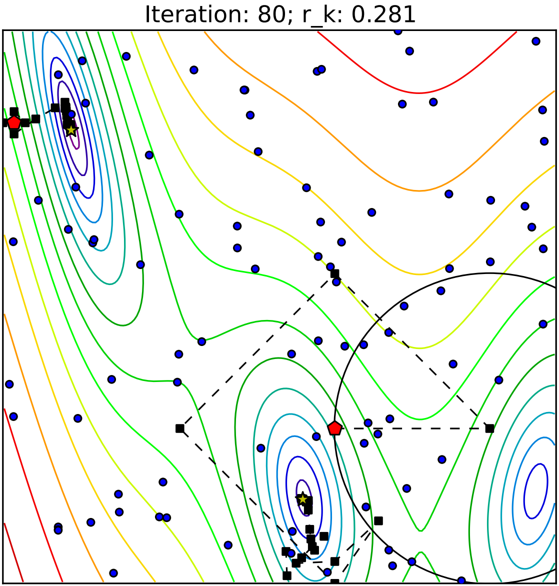
BAMLM



BAMLM

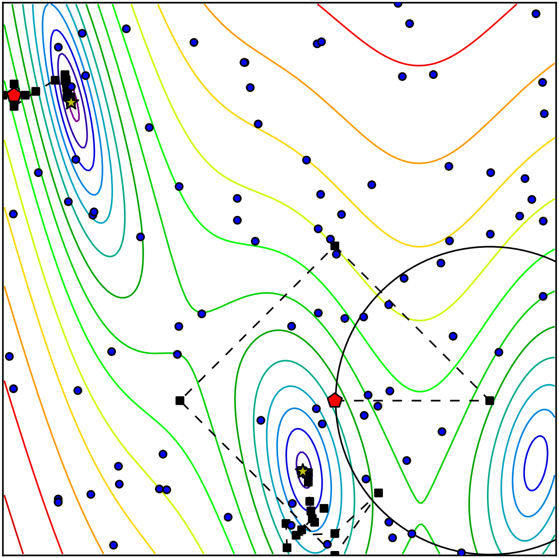


BAMLM

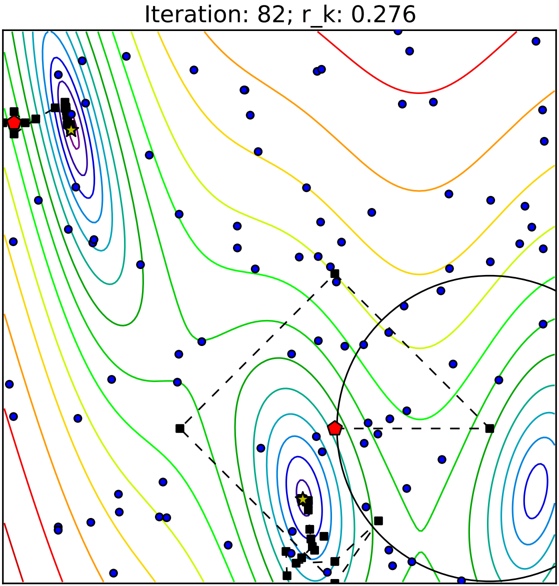


BAMLM

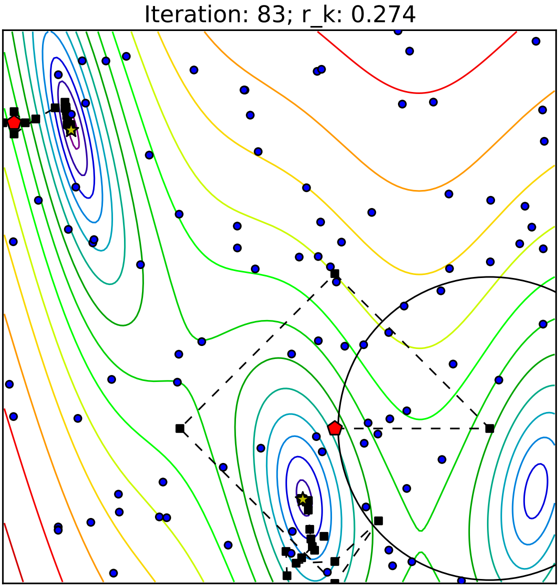
Iteration: 81; r_k: 0.279



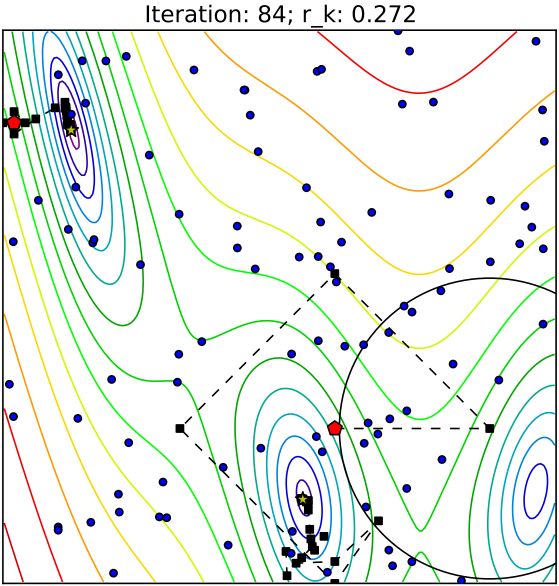
BAMLM



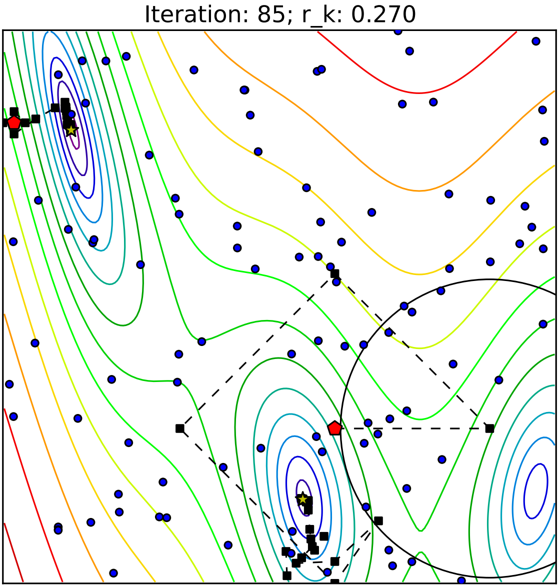
BAMLM



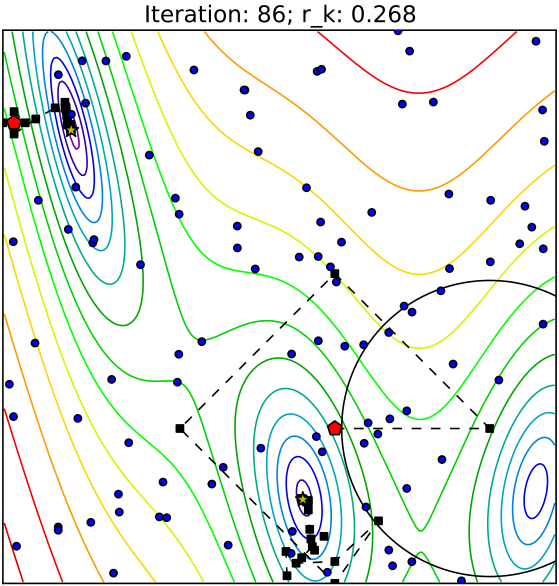
BAMLM



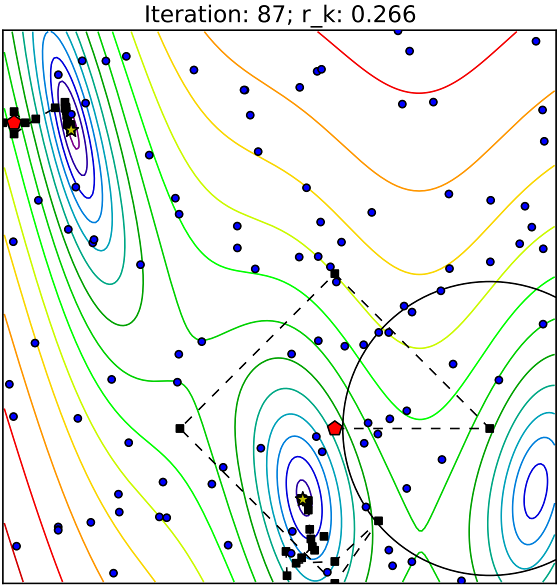
BAMLM



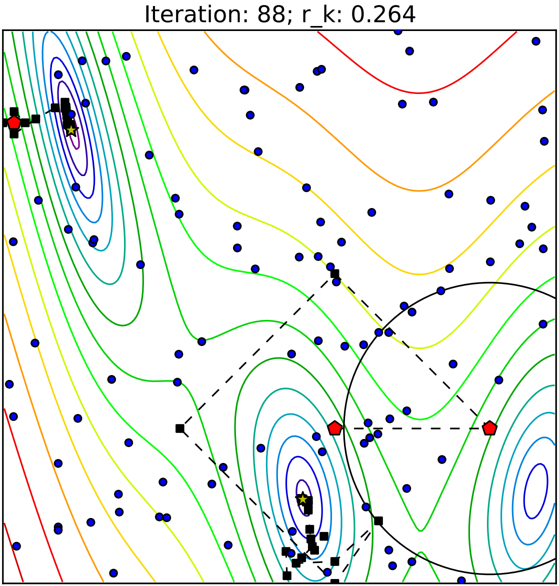
BAMLM



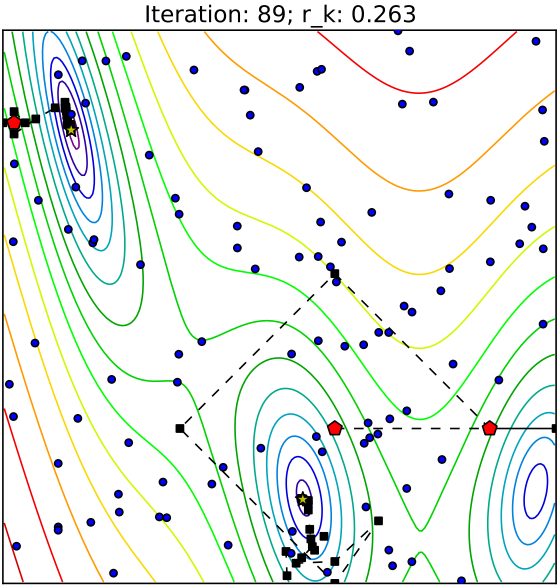
BAMLM



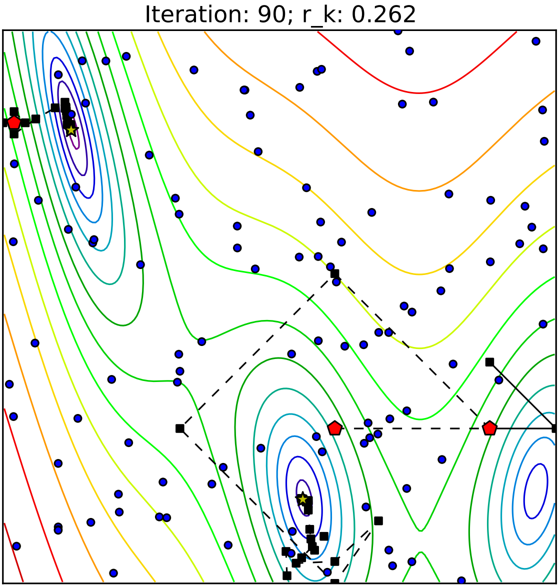
BAMLM



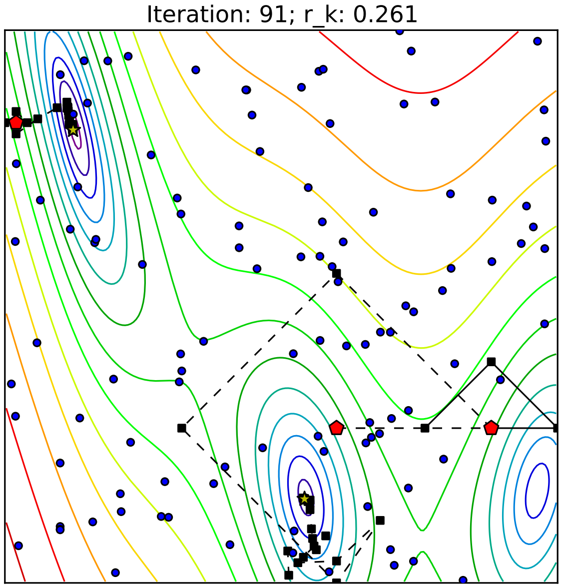
BAMLM



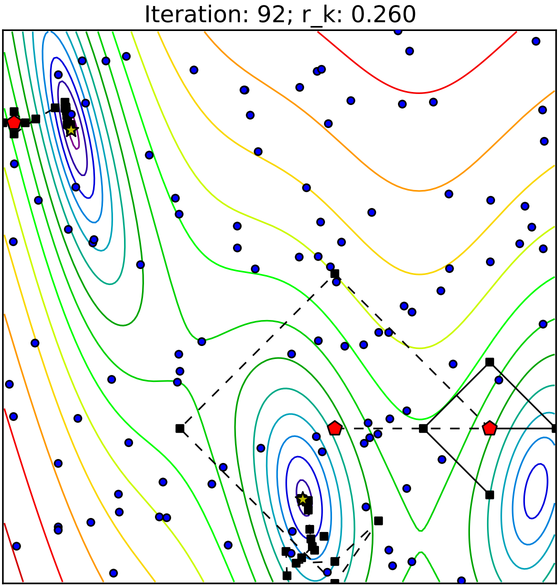
BAMLM



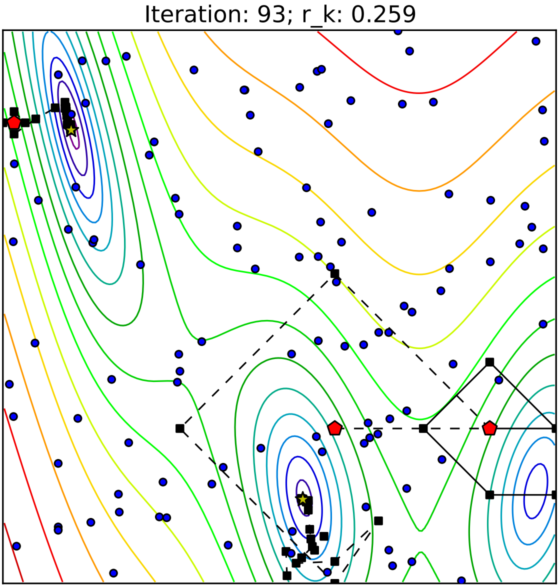
BAMLM



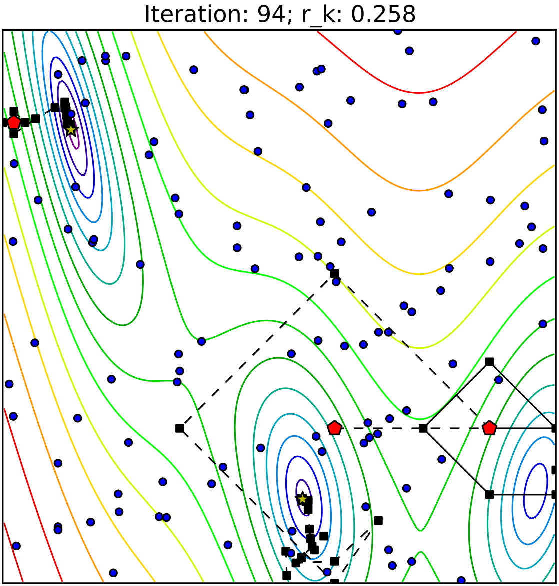
BAMLM



BAMLM

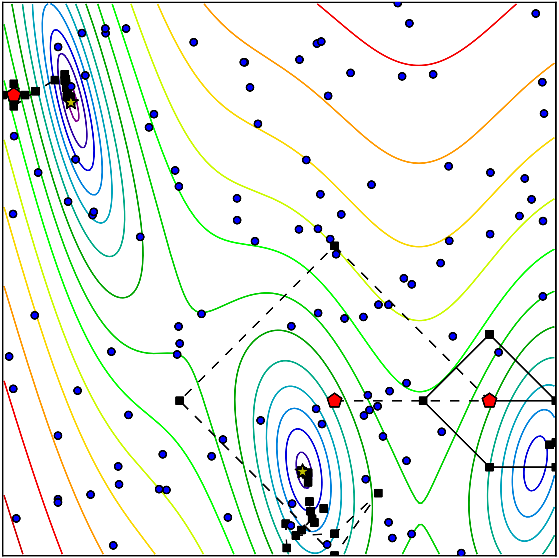


BAMLM

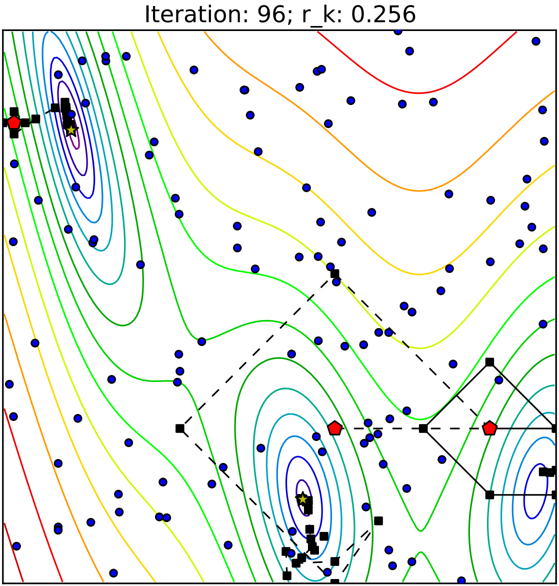


BAMLM

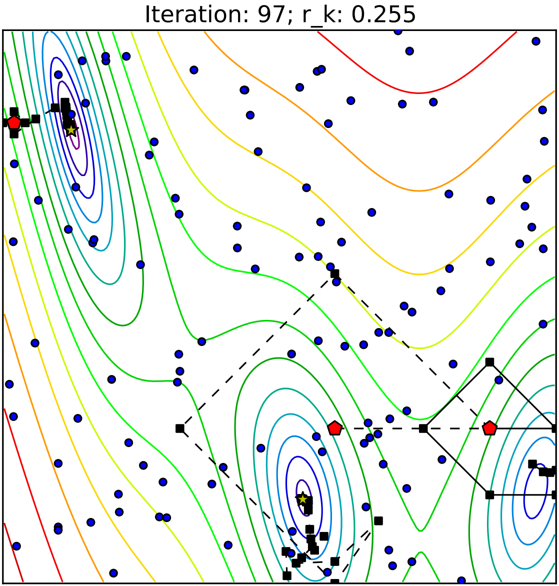
Iteration: 95; r_k: 0.257



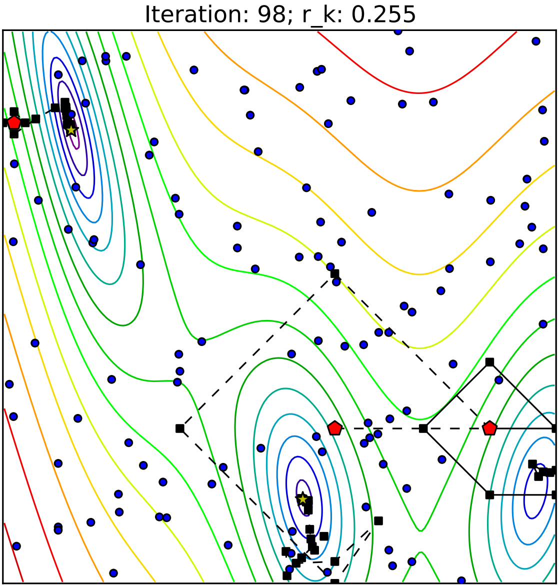
BAMLM



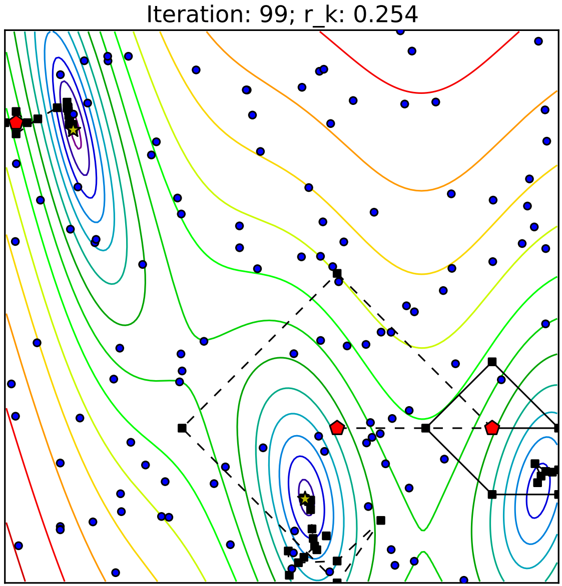
BAMLM



BAMLM



BAMLM



Properties of the local optimization method

Necessary:

- Honors a starting point
- Honors bound constraints

Properties of the local optimization method

Necessary:

- Honors a starting point
- Honors bound constraints

ORBIT satisfies these [Wild, Regis, Shoemaker, SIAM-JOSC, 2008]

BOBYQA satisfies these [Powell, 2009]

Properties of the local optimization method

Necessary:

- Honors a starting point
- Honors bound constraints

ORBIT satisfies these [Wild, Regis, Shoemaker, SIAM-JOSC, 2008]

BOBYQA satisfies these [Powell, 2009]

Possibly beneficial:

- Can return multiple points of interest
- Reports solution quality/confidence at every iteration
- Can avoid certain regions in the domain
- Uses a history of past evaluations of f
- Uses additional points mid-run

AAMLM

Algorithm 2: AAMLM

Give each worker a point to evaluate
for $k = 1, 2, \ldots$ **do**
  Receive from (longest waiting) worker w that has evaluated f
  Update $\mathcal{H}_k$ and r_k
  if *point evaluated by* w *is from an active run* **then**
    if *Run is complete* **then**
      Update X_k^*, and mark points inactive
    else
      Add the next point in its localopt run (not in $\mathcal{H}_k$) to Q_L
    Start run(s) at all point(s) satisfying (S1)–(S4), (L1)–(L6)
    Add the subsequent point (not in $\mathcal{H}_k$) from each run to Q_L
  Merge runs in Q_L with candidate minima within 2ν of each other
  Give w a point at which to evaluate f, either from Q_L or $\mathcal{R}$

BAMLM

MLSL: (S2)–(S4)

$\hat{x} \in \mathcal{S}_k$

(S1) $\nexists x \in \mathcal{L}_k$ with $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

(S2) *$\nexists x \in \mathcal{S}_k$ with $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$*

(S3) *$\hat{x}$ has not started a local optimization run*

(S4) *$\hat{x}$ is at least μ from $\partial \mathcal{D}$ and ν from known local minima*

BAMLM: (S1)–(S4), (L1)–(L6)

$\hat{x} \in \mathcal{L}_k$

(L1) $\nexists x \in \mathcal{L}_k$ $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

(L2) $\nexists x \in \mathcal{S}_k$ with $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

(L3) $\hat{x}$ has not started a local optimization run

(L4) $\hat{x}$ is at least μ from $\partial \mathcal{D}$ and ν from known local minima

(L5) $\hat{x}$ is not in an active local optimization run and has not been ruled stationary

(L6) $\exists r_k$-descent path in $\mathcal{H}_k$ from some $x \in \mathcal{S}_k$ satisfying (S2-S4) to $\hat{x}$

AAMLM Theory

Theorem

Given the same assumptions as MLSL, AAMLM will start a finite number of local optimization runs with probability 1.

AAMLM Theory

Theorem

Given the same assumptions as MLSL, AAMLM will start a finite number of local optimization runs with probability 1.

Assumption

There exists $K_0 < \infty$ so that for any K_0 consecutive iterations, there is a positive (bounded away from zero) probability of evaluating a point from the sample stream and each existing local optimization run.

AAMLM Theory

Theorem

Given the same assumptions as MLSL, AAMLM will start a finite number of local optimization runs with probability 1.

Assumption

There exists $K_0 < \infty$ so that for any K_0 consecutive iterations, there is a positive (bounded away from zero) probability of evaluating a point from the sample stream and each existing local optimization run.

Theorem

Each $x^* \in X^*$ will almost surely be either identified in a finite number of evaluations or have a single local optimization run that is converging asymptotically to it.

Measuring Performance

GLODS Global & local optimization using direct search [Custódio, Madeira (JOGO, 2014)]

Direct Serial DIRECT [D. Finkel's MATLAB code]

pVTDirect Parallel DIRECT [He, Watson, Sosonkina (TOMS, 2009)]

Random Uniform sampling over domain (as a baseline)

BAMLM

- Concurrency: 4
- Local optimization method
  - ORBIT [Wild, Regis, & Shoemaker (SIAM JOSC, 2008)]
  - BOBYQA [Powell, 2009]
- Initial sample size: $10n$

- Each method evaluates Direct's $2n+1$ initial points.

Measuring Performance

Notation:

Let X^* be the set of all local minima of f.
Let $f^*_{(i)}$ be the ith smallest value $\{f(x^*)|x^* \in X^*\}$.
Let $x^*_{(i)}$ be the element of X^* corresponding to the value $f^*_{(i)}$.

The global minimum has been found at a level $\tau > 0$ at batch k if an algorithm it has found a point $\hat{x}$ satisfying:

$$f(\hat{x}) - f^*_{(1)} \leq (1-\tau)\left(f(x_0) - f^*_{(1)}\right),$$

where x_0 is the starting point for problem p.

Measuring Performance

Notation:

Let X^* be the set of all local minima of f.
Let $f^*_{(i)}$ be the ith smallest value $\{f(x^*)|x^* \in X^*\}$.
Let $x^*_{(i)}$ be the element of X^* corresponding to the value $f^*_{(i)}$.

The j best local minima have been found at a level $\tau > 0$ at batch k if:

$$\left|\left\{x^*_{(1)}, \ldots, x^*_{(\underline{j}-1)}\right\} \bigcap \left\{x^*_{(i)} : \exists x \in \mathcal{H}_k \text{ with } \left\|x - x^*_{(i)}\right\| \leq r_n(\tau)\right\}\right| = \underline{j} - 1$$

$$\&$$

$$\left|\left\{x^*_{(\underline{j})}, \ldots, x^*_{(\bar{j})}\right\} \bigcap \left\{x^*_{(i)} : \exists x \in \mathcal{H}_k \text{ with } \left\|x - x^*_{(i)}\right\| \leq r_n(\tau)\right\}\right| \geq j - \underline{j} + 1,$$

where $\underline{j}$ and $\bar{j}$ are the smallest and largest integers such that

$f^*_{(\bar{j})} = f^*_{(j)} = f^*_{(\underline{j})}$ and where $r_n(\tau) = \sqrt[n]{\frac{\tau \operatorname{vol}(\mathcal{D})\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$.

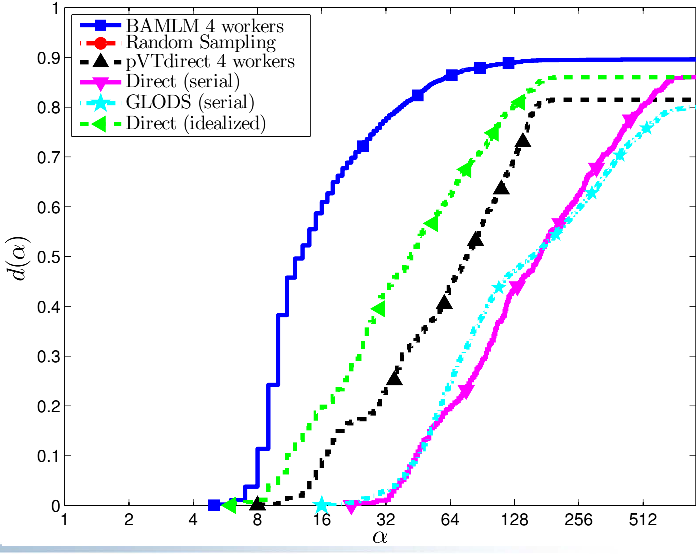
Problems considered

GKLS problem generator [Gaviano et al., "Algorithm 829" (TOMS, 2003)]

- 600 synthetic problems with known local minima
- $n = 2, \ldots, 7$
- 10 local minima in the unit cube with a unique global minimum
- 100 problems for each dimension
- 5 replications (different seeds) for each problem
- 5000 evaluations

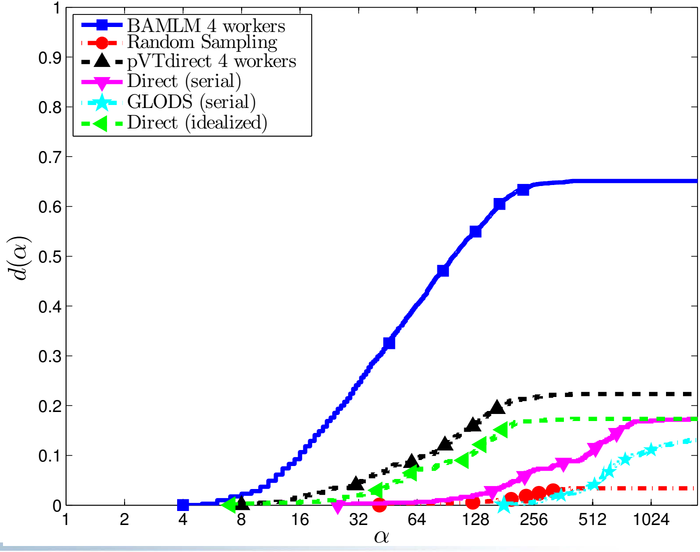
Data Profiles

$$f(x) - f^*_{(1)} \leq (1 - 10^{-5}) \left(f(x_0) - f^*_{(1)} \right)$$



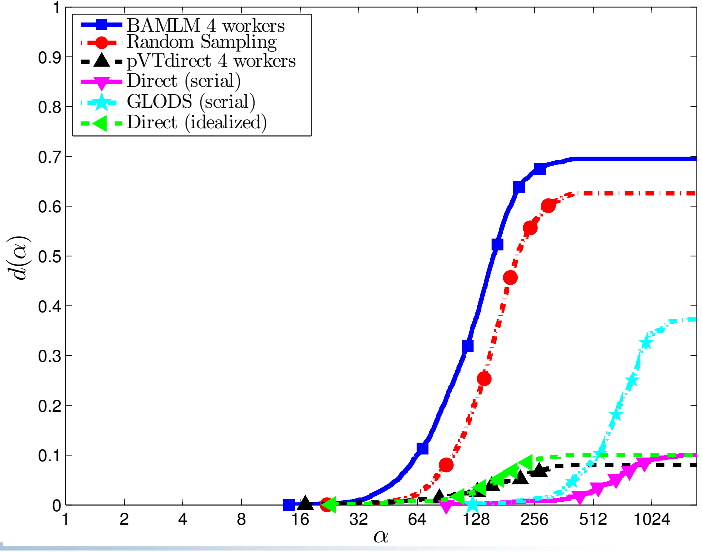
Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 3 best minima



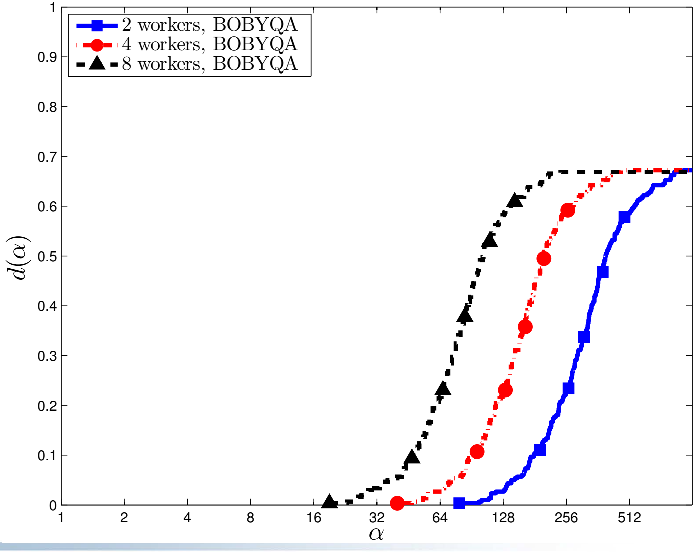
Data Profiles

Within $\sqrt[n]{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 7 best minima



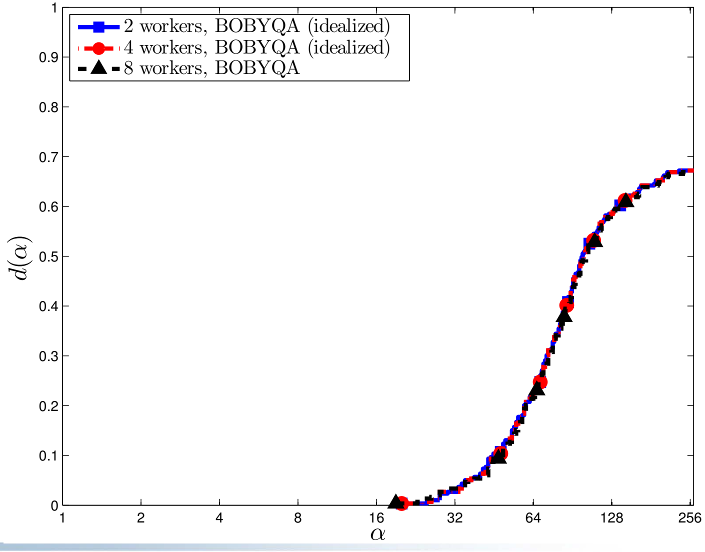
Data Profiles

Within $\sqrt[n]{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 7 best minima

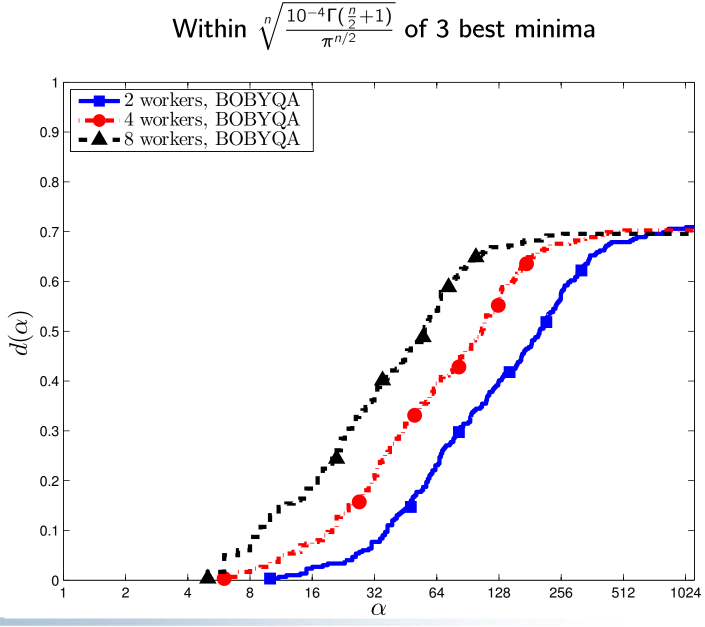


Data Profiles

Within $\sqrt[n]{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 7 best minima

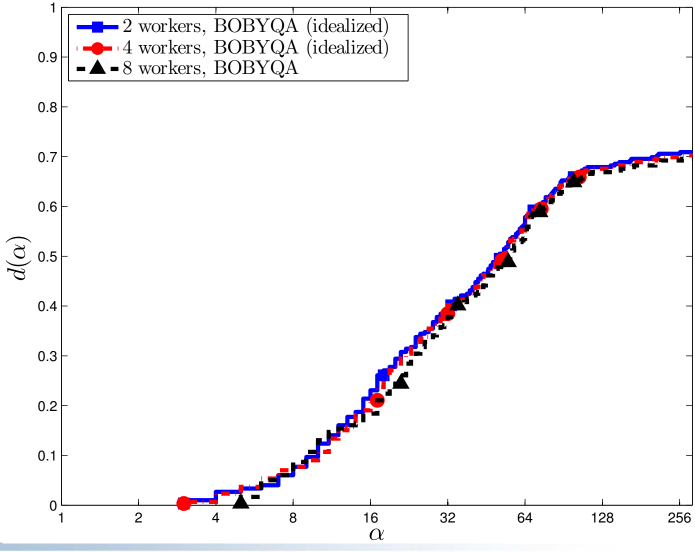


Data Profiles



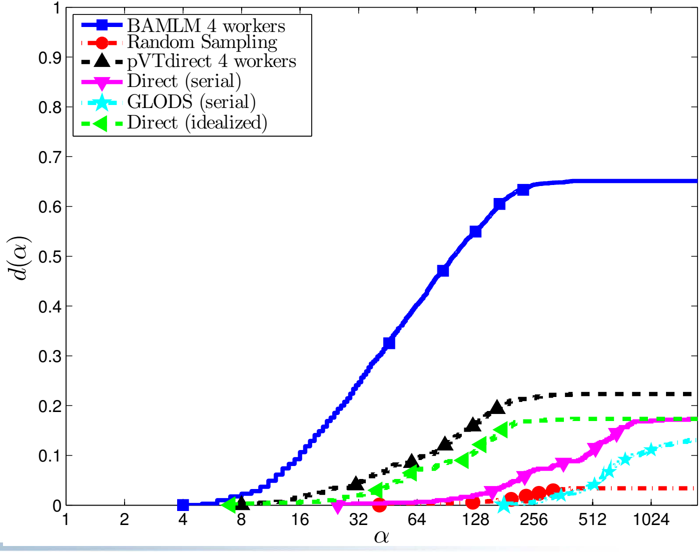
Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 3 best minima



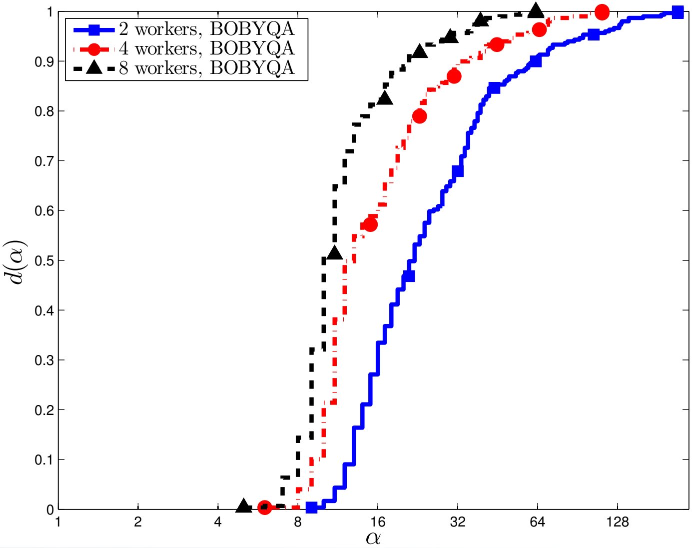
Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 3 best minima



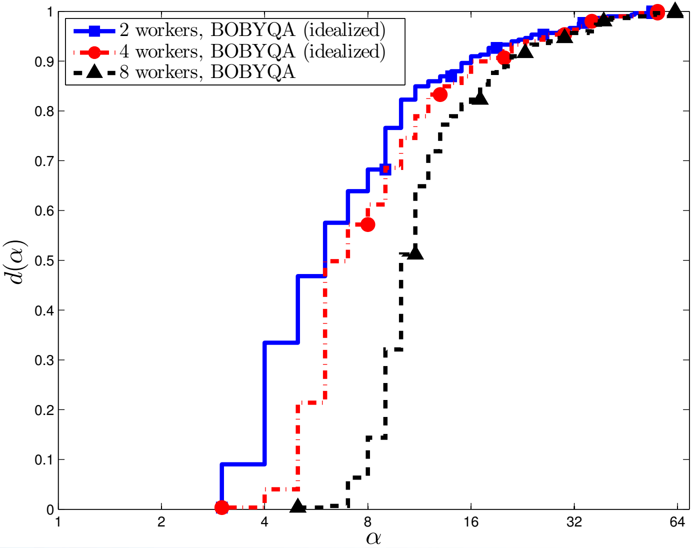
Data Profiles

$$f(x) - f^*_{(1)} \leq (1 - 10^{-5}) \left(f(x_0) - f^*_{(1)} \right)$$



Data Profiles

$$f(x) - f^*_{(1)} \leq (1 - 10^{-5}) \left(f(x_0) - f^*_{(1)} \right)$$



Closing Remarks

- Concurrent function evaluations can locate multiple minima while efficiently finding a global minimum.

Closing Remarks

- Concurrent function evaluations can locate multiple minima while efficiently finding a global minimum.

- Write/use algorithms that exploit problem structure

Closing Remarks

- Concurrent function evaluations can locate multiple minima while efficiently finding a global minimum.

- Write/use algorithms that exploit problem structure

Current work:

- Finding (or designing) the best local solver for our framework?
- Best way to process the queue?

AAMLM

Algorithm 2: AAMLM

Give each worker a point to evaluate
for $k = 1, 2, \ldots$ **do**
    Receive from (longest waiting) worker w that has evaluated f
    Update $\mathcal{H}_k$ and r_k
    if *point evaluated by* w *is from an active run* **then**
        if *Run is complete* **then**
            Update X_k^*, and mark points inactive
        else
            Add the next point in its localopt run (not in $\mathcal{H}_k$) to Q_L
        Start run(s) at all point(s) satisfying (S1)–(S4), (L1)–(L6)
        Add the subsequent point (not in $\mathcal{H}_k$) from each run to Q_L
    Merge runs in Q_L with candidate minima within 2ν of each other
    Give w a point at which to evaluate f, either from Q_L or $\mathcal{R}$